J. M. Johnston

MICROS

BIPS, PITIDOS Y LUCES

Libro segundo

500 pt.

MICROS: BIPS, PITIDOS Y LUCES

Micros: Bips, pitidos y luces

J. M. Johnston

Ilustraciones de Len Epstein

ANAYA MULTIMEDIA

MICROINFORMATICA

Título de la obra original:
FIRST LOOK COMPUTERS 2: COMPUTERS: BEEPS, WHIRS AND BLINKING LIGHTS

Traducción de: Pilar Vázquez
Diseño de colección: Antonio Lax
Diseño de cubierta: José Lorente
Ilustraciones: Len Epstein

Reservados todos los derechos. Ni la totalidad ni parte de este libro puede reproducirse o transmitirse por ningún procedimiento electrónico o mecánico, incluyendo fotocopia, grabación magnética o cualquier almacenamiento de información y sistema de recuperación, sin permiso escrito de Ediciones Anaya Multimedia, S. A.

Copyright © 1983 by J. M. Johnston

PUBLISHED BY ARRANGEMENT WITH DELL PUBLISHING CO., INC., NEW YORK, N.Y. and BANBURY BOOKS, INC., WAYNE, P.A., U.S.A.

© EDICIONES ANAYA MULTIMEDIA, S. A., 1985
Villafranca, 22. 28028 Madrid
Depósito legal: M. 21.153-1985
ISBN: 84-7614-025-8
Printed in Spain
Imprime: HEROES, S. A. Torrelara, 8. 28016 Madrid

Qué estás haciendo?

Estoy haciendo un trabajo para el colegio acerca de todas las cosas que he aprendido sobre los ordenadores.

¿Y qué es lo que has aprendido?

¡Oh! Muchísimas cosas. Todo el mundo utiliza los ordenadores: los médicos, en los hospitales; la policía; los pilotos, en los aviones..., y otras muchas personas.

Se emplean en las fábricas, para controlar las temperaturas y las máquinas; en las escuelas, para enseñar a los alumnos, y en los bancos, para llevar las cuentas de los clientes.

También se usan en el Metro, en las oficinas de Correos, en las bibliotecas, y en...

¿Y explicas en el trabajo cómo funcionan los ordenadores?

Bueno, digo que los ordenadores son como el cerebro humano, porque pueden sumar y restar, tomar decisiones y controlar cosas. Utilizan una cosa que se llama entrada, *al igual que nosotros utilizamos los ojos y los oídos; la información que nos dan se llama* salida, *y equivale a las cosas que nosotros decimos o escribimos. También tienen* memoria, *para guardar la información.*

Pero ahora resulta que tengo un problema.

¿Cuál?

Bueno, a lo mejor deberíamos darnos una vuelta por dentro de uno y ver qué es lo que pasa.

Esperaba que me lo propusieras. ¿Por dónde empezamos?

Vayamos directamente al corazón del asunto: la CPU.

Ya sabes que te voy a preguntar qué es la CPU.

Me desilusionarías si no lo hicieras. La CPU es donde tiene lugar todo lo que es pensamiento real en el ordenador. Son las siglas de *Central Processing Unit* (Unidad Central de Proceso) y es la parte del ordenador más parecida a tu cerebro.

Me imagino, entonces, que todos los ordenadores tienen que tener una CPU.

Eso es. La CPU es la parte más importante del sistema, al igual que tu cerebro es la parte más importante de tu cuerpo. Sin él no puedes hacer nada.

¿Y esa CPU está en el mismo sitio que la memoria del ordenador?

No exactamente. En tu cerebro está todo junto en un solo bloque, pero en los ordenadores las cosas suelen estar en bloques separados.

No obstante, la CPU y la memoria están bastante relacionadas, pues la CPU, para llevar a cabo su trabajo, utiliza la información almacenada en la memoria del ordenador.

¿Y qué tipo de trabajo hace?

La CPU se encarga de todas esas cosas de las que estabas hablando hace un momento: la aritmética, el control, la toma de decisiones... Normalmente, está dividida en dos partes: una parte se ocupa de la aritmética y las decisiones, y la otra, del control.

En primer lugar, hablaremos de la aritmética. Lo que esta parte hace principalmente es contar y sumar. También resta, multiplica y divide, por supuesto, además de otras muchas operaciones aritméticas, pero contar y sumar son las más importantes.

Para comprender cómo cuenta y suma la CPU es importante recordar lo que tú y yo usamos cuando contamos y sumamos.

Eso está tirado. Usamos números.

Eso es. Y un número está compuesto de símbolos. ¿Sabes lo que es un *símbolo?*

En la escuela nos han dicho que un símbolo es aquello que está en representación de algo.

Dame un ejemplo.

Bueno... por ejemplo, una bandera es un símbolo, porque representa un país.

Bien. Cuando ves la bandera española piensas en España.
Aquí tienes más símbolos, ¿puedes decirme lo que representan?

Una iglesia..., una barbería..., paz... y «No fumar».

Vale. ¿Y qué me dices de este otro?

¿Puede significar algo más? ¿Qué me dices del número romano para 10?

También podría ser sencillamente la letra X del alfabeto.

Y para los ingleses y americanos, varias X juntas significan besos y abrazos.

Algunos símbolos tienen, entonces, más de un significado, ¿no? Pues, en los ordenadores, los símbolos también pueden tener diferentes significados, dependiendo de cómo se utilicen.

Ahora mira, los números también están compuestos de símbolos. ¿Cuántos símbolos numéricos diferentes empleamos cuando contamos?

¡Oh, muchos! Miles.

No, sólo diez.

¿Sólo diez?

Claro. Y son éstos:

0 1 2 3 4 5 6 7 8 9

Todos los números que usamos están compuestos por tan sólo estos diez símbolos, colocados de diferentes maneras:

17 2486 0 999 8937702119

¿Por qué crees que utilizamos diez símbolos en lugar de cinco o veinte o cualquier otra cantidad?

Pues, no lo sé.

¿Cuántos dedos tienes?

¡Diez!

Como el hombre primitivo seguramente usaba los dedos para ayudarse a contar, al igual que lo hacen los niños pequeños, probablemente continuó usándolos cuando empezó a escribir los números. Incluso los numerales romanos aparecen en grupos de diez.

Los primeros constructores de ordenadores también intentaron utilizar diez símbolos. Emplearon unos engranajes con diez dientes que tenían que girar hasta ponerse en la posición adecuada en el caso de cada número.

Pero los engranajes eran muy lentos, y no paraban de atascarse. La CPU de los ordenadores de hoy utiliza *electricidad* para contar, en lugar de engranajes. La electricidad es extremadamente rápida, y tan sólo utiliza dos «dedos».

¡Huy! ¿Qué quieres decir con eso de dos dedos?

Bueno, piensa en todos los sencillos aparatos eléctricos que hay en tu casa, como las bombillas o los ventiladores. Lo único que haces es *encenderlos* y *apagarlos,* ¿no?

Pues, sí.

Y para hacerlo utilizas un interruptor. Pues dentro de la CPU hay muchísimos interruptores pequeñitos, tan pequeños que necesitarías un microscopio para verlos.

Cada uno de estos interruptores puede encenderse y apagarse muy rápidamente, millones de veces por *segundo.* Y puesto que sólo hay dos posibles posiciones, la CPU usa el símbolo 1, para *encendido,* y el símbolo 0, para *apagado.*

¿Pero cómo puede contar y resolver problemas aritméticos con tan sólo el 0 y el 1?

Del mismo modo que nosotros lo hacemos con diez símbolos. Veámoslo detenidamente.

En nuestro sistema, que se llama *decimal* porque en latín «decem» significa diez, nosotros utilizamos los símbolos del 0

al 9; pero cada vez que pasamos del 9 nos quedamos sin símbolos, ¿no es verdad?

Así que entonces empezamos a unir los símbolos de dos en dos, en parejas. El número 15, por ejemplo, se compone de un 1 y un 5 puestos juntos. De este modo podemos escribir los números del 10 al 99. ¿Pero qué pasa cuando llegamos al 99?

Volvemos a quedarnos sin símbolos.

Así que empezamos a ponerlos de tres en tres. Con esto tenemos números desde el 100 al 999.

Y luego unimos los símbolos de cuatro en cuatro, para los números comprendidos entre el 1000 y el 9999.

Eso es. Cada vez que añadimos un símbolo hacemos números más y más largos. Pasamos de las *unidades* a las *decenas*, y de ahí a las *centenas*, y después a las *unidades de millar*, y así sucesivamente.

De este modo formamos todos los números que podamos llegar a necesitar, con tan sólo diez símbolos originales.

Del mismo modo. Cada vez que se queda sin símbolos, añade más ceros y unos para hacer números más grandes.

¡Pero si se queda sin símbolos después del «1»!

Eso es cierto. Sencillamente utiliza un par de símbolos para hacer el número 2 que se escribe «10», y que, aunque parece «diez», en el ordenador representa al número 2.

Y 3 se escribe «11». Pero en este momento a la CPU se le agotan las posibilidades de unir los símbolos por parejas, de modo que tiene que empezar a utilizar tres símbolos. El número 4, por ejemplo, se escribe «100».

Hay varias maneras de poner tres símbolos juntos, como «101» para 5 y «110» para 6. Pero más pronto o más tarde tiene que empezar a utilizar cuatro símbolos: el número 8 se escribe «1000». Y luego cinco símbolos: el número 16 es «10000», y así sucesivamente.

Y así es como la CPU, utilizando solamente los símbolos 0 y 1, forma todos los números que necesita para contar. Este sistema se llama *binario,* porque «bi» significa «dos», como la bicicleta, que tiene dos ruedas.

Pero la CPU no se limita a contar, sino que hace más cosas, ¿no? ¿Puede sumar y restar con esos números binarios?

Ciertamente; igual que lo hacemos nosotros con los números decimales. Nosotros escribimos:

```
  8
+ 6
---
 14
```

En números binarios escribiríamos:

```
  1000 (que representa 8)
+  110 (que representa 6)
------
  1110 (que representa 14)
```

Claro que la cosa se complica un poco cuando tienes que restar o multiplicar. Pero a la CPU no le importa nada; simplemente acciona esos pequeños interruptores sumando y restando a una velocidad tal que te daría vértigo.

Pero no tienes por qué imaginártelas; el ordenador siempre escribe los números en el sistema decimal o en otro tipo de sistema que la gente esté acostumbrada a usar.

¿Quieres decir que hay otros sistemas numéricos?

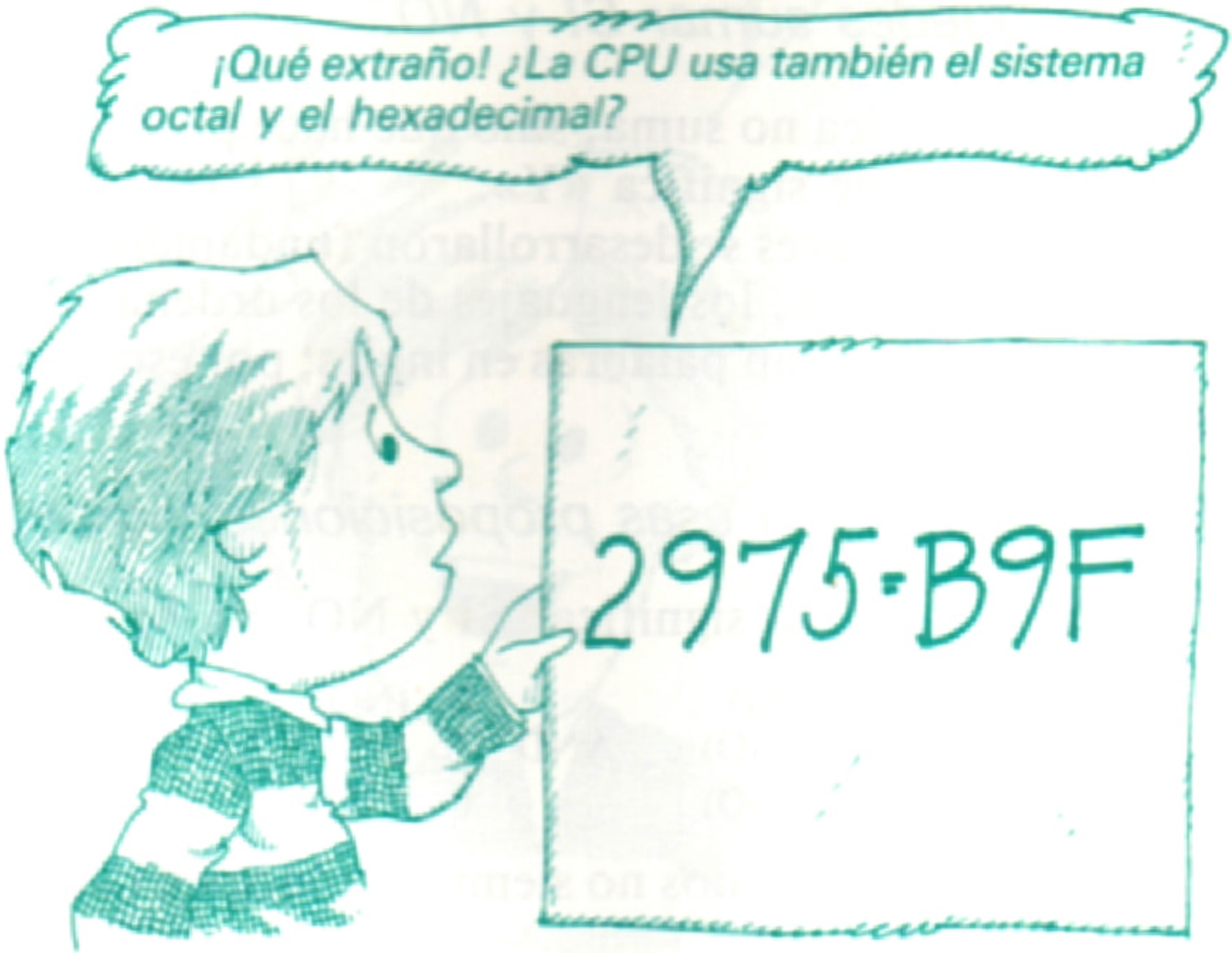
Por supuesto. Hay el *octal*, que utiliza *ocho* símbolos, del 0 al 7; y también el *hexadecimal*, que utiliza *dieciséis* símbolos.

¿Dieciséis? Creía que habías dicho que no había más símbolos después del 9.

Y no los hay; así que se utilizan letras del alfabeto para representar el 10, 11, 12, 13, 14, 15, con lo que los símbolos hexadecimales son:

0 1 2 3 4 5 6 7 8 9 A B C D E F

Por ejemplo, el número decimal 2975 se escribiría así en hexadecimal: «B9F».



No; la CPU utiliza sólo números binarios para llevar a cabo todas las operaciones aritméticas y lógicas.

¿Qué quiere decir «lógicas»?

Es el tipo de trabajo que realiza la CPU cuando toma decisiones.

¿Pero qué tipo de decisiones puede tomar la CPU con sólo dos símbolos?

En seguida lo vas a entender, porque es muy fácil. El tomar una decisión significa hacerte una pregunta, ¿no? ¿Debo hacer esto o debo hacer aquello? Y a cada posibilidad puedes contestar con un sí o con un no. Sí, debo hacer esto..., o no, no debo hacer aquello.

¿Recuerdas que dijimos que en los ordenadores los símbolos solían tener significados diferentes? Bueno, pues digamos que, en la sección de lógica de la CPU, el símbolo 1 representa SI y el símbolo 0 representa NO.

Pero no puedes sumar SI y NO.

La sección de lógica no suma, sino que hace proposiciones lógicas con AND, que significa «Y».

Como los ordenadores se desarrollaron fundamentalmente en países angloparlantes, los lenguajes de los ordenadores, el BASIC, por ejemplo, usan palabras en inglés; por eso decimos «AND», «THEN», «IF»...

¿Y cómo se hacen esas proposiciones lógicas?

Utilizando 1 y 0 para significar SI y NO. En la CPU,

1 (SI)	1 (SI)	0 (NO)	0 (NO)
AND 1 (SI)	AND 0 (NO)	AND 1 (SI)	AND 0 (NO)
1 (SI)	0 (NO)	0 (NO)	0 (NO)

Observa que los resultados no siempre coinciden con los de las sumas.

No entiendo nada.

Bueno, pongamos un ejemplo. Imagínate que en tu colegio le fueran a dar un premio al que pudiera responder SI a estas *dos* preguntas:

1. ¿Tienes un diez en todas las asignaturas?
2. ¿Has tenido un comportamiento perfecto?

Si tus respuestas fueran SI y SI, es decir, 1 y 1, el premio sería tuyo, ya que si vuelves a mirar el gráfico verás que 1 AND 1 nos da 1, y 1 representa SI.

Si *una* de tus respuestas fuera NO, tendríamos 1 AND 0 (o 0 AND 1), lo que nos da 0, que representa NO, y entonces no tendrías el premio.

Y si tus dos respuestas fueran NO, es decir, 0 AND 0, que también nos da 0, está claro que tampoco conseguirías el premio.

Sí, claro; se pueden hacer proposiciones lógicas con OR (que significa «O»).

¿Y cómo se hacen?

Bueno, 1 sigue siendo SI y 0 sigue siendo NO, pero en este caso:

1 (SI)	1 (SI)	0 (NO)	0 (NO)
OR 1 (SI)	OR 0 (NO)	OR 1 (SI)	OR 0 (NO)
1 (SI)	1 (SI)	1 (SI)	0 (NO)

Suponte ahora que todo lo que necesitaras para ganar el premio fuera poder responder SI a *una* de esas dos mismas preguntas. Podrías tener un diez en todas las asignaturas O un comportamiento perfecto; no es necesario tener los dos.

Si respondieras SI y SI, es decir, 1 OR 1, que nos da 1, ganarías el premio.

Si una de tus respuestas fuera NO, es decir, 1 OR 0 (o 0 OR 1), como nos sigue dando 1, también tendrías premio.

Si tus respuestas fueran las dos NO, como 0 OR 0 nos da 0, y 0 representa NO, no tendrías premio.

Eso es verdad. Pero al ordenador le dan igual los premios. Utiliza las proposiciones para decidir cosas tales como si ha de sumar o restar y qué debe hacer con las respuestas.

¿Cómo sabe la CPU qué símbolos ha de sumar y cuáles otros ha de utilizar para hacer proposiciones lógicas?

Buena pregunta. La respuesta es que la sección de control se lo dice.

La sección de control de la CPU es como esa parte de tu cerebro que te dice que pongas un pie delante de otro cuando estás andando, en qué dirección has de mover el brazo cuando te lavas los dientes, o cómo has de mover la boca y la lengua al hablar.

Ya; y lo que hace la CPU continuamente es *procesar:* recuerda que CPU son las siglas de *Central Processing Unit* (Unidad Central de *Proceso).* Así que en la CPU las secciones de

aritmética y lógica se encargan del procesado, y la sección de control les dice *cómo* y *cuándo* hacerlo.

Ahora hemos de tener en cuenta otras dos cuestiones que todavía no hemos tocado: *¿qué* es lo que se procesa?, *¿dónde* está?

¡Yo sé la respuesta a esas dos preguntas! La CPU procesa información *y la información está en la* memoria *del ordenador.*

Exacto. Conque ahora lo que tendríamos que hacer es echar un vistazo a la memoria. Y como la memoria está en una parte diferente del ordenador, tendremos que coger un *bus*.

¿*Un* bus?

Sí, eso es. Toda la información que la memoria envía a la CPU, y que ésta vuelve a enviar a la memoria, viaja en una cinta de cables lisa que se llama *bus*. También hay *buses* que viajan entre la memoria y los *puertos* de entrada y salida.

Casi. En el ordenador, el *puerto* es el lugar donde la información proveniente de los mecanismos de entrada «atraca» an-

tes de tomar un *bus* hasta la memoria. Y también donde la información que sale deja el *bus* para ir a uno de los mecanismos de salida. ¿Sabes cuáles son esos mecanismos de entrada y salida?

Pues hay terminales..., y sensores..., y discos, y cintas.

Bien. Así que, como ves, la memoria de un ordenador se parece a una gran estación de autobuses, en la que, sin cesar, está llegando nueva información que pasa a otros *buses* y parte seguidamente.

¡Parece que es un sitio muy ajetreado!

Y lo es. Y normalmente también muy repleto. Por esta razón, la mayor parte de la información no se queda allí mucho tiempo. Está impaciente por llegar a su destino y dejar sitio a la nueva información que está esperando para entrar.

¿Cómo es la memoria?

Bueno, hay memorias de varios tipos; pero la mayoría de ellas utilizan hoy esos pequeños interruptores que vimos en la CPU. Hay miles de ellos en cada memoria, y se encienden y se apagan a toda velocidad, según va pasando por ellos la electricidad.

Los interruptores de la memoria se llaman *células*, como las células del cerebro en las que se guarda toda la información cerebral. Y las células de la memoria del ordenador también almacenan información. Cada célula puede contener exactamente un símbolo.

Eso es. Si el interruptor está encendido, significa que la electricidad está pasando por él, y la célula contiene un 1; cuando el interruptor está apagado, significa que no pasa la electricidad, y la célula contiene un 0.

¿Significa eso, entonces, que toda la información almacenada en las células de la memoria sólo puede usarse para resolver problemas aritméticos y tomar decisiones?

No. Los símbolos binarios de la memoria también pueden representar otras cosas, como las letras del alfabeto, las frases, las comas, los acentos... y todas las cosas que puedes encontrar en una máquina de escribir o en el teclado de un terminal.

Y *esos* símbolos también pueden unirse para formar palabras, y nombres, y direcciones, e incluso dibujos. De poco nos serviría el ordenador si sólo pudiera manejar números, ¿no?

Sí, algo así. No podrías leer libros, ni dibujar, ni siquiera hablar.

Por tanto, sabemos que las células de la memoria del ordenador contienen símbolos binarios que representan muchas clases diferentes de información.

¿Pero cómo sabe el ordenador qué tipo de información representan todos esos ceros y unos?

No lo sabe. Hay que decírselo.

¿Y quién se lo dice?

Un *programa* se encarga de ello. El programa es como las instrucciones que te dan con los juegos de construcciones o como las recetas que se utilizan para cocinar. Va mostrando paso a paso cómo se supone que se deben hacer las cosas.

¿Pero dónde está el programa?

Allí mismo, en la memoria. Forma parte de la entrada, que se introduce en el ordenador mediante un terminal o un disco. Y queda almacenado en la memoria hasta que la CPU tenga que usarlo. A ver si adivinas qué parte de la CPU usa las instrucciones del programa para llevar a cabo sus funciones.

La sección de control.

Correcto. Una vez que el programa ha quedado almacenado en la memoria, la sección de control de la CPU se pone a trabajar. Va recogiendo de una en una las instrucciones del programa, mira lo que dice que hay que hacer y escoge la parte aritmética o lógica adecuada para hacerlo. Luego vuelve a la memoria para recoger la siguiente instrucción.

Si la instrucción dice que *sume* dos números, buscará esos números en la memoria y los enviará a la sección de aritmética. Si la instrucción dice que los *compare*, los enviará a la sección de lógica. Si la orden dice que *repita* alguna de las instrucciones dadas, vuelve y busca en la memoria esas instrucciones para ejecutarlas de nuevo.

Y todos los números y cosas que el programa le dice a la CPU que use están también en la memoria, ¿verdad?

Sí, por supuesto. También forman parte de la entrada del ordenador. Además tienen un nombre: se les llama *datos*.

Así pues, el programa le dice a la CPU qué datos tiene que usar y qué es lo que ha de hacer con ellos. Tanto los programas como los datos constituyen la *entrada* del ordenador, y ambos están almacenados en la *memoria*. Más adelante veremos cómo se escribe un programa y cómo éste utiliza los datos.

¿Puede haber más de un programa al mismo tiempo en la memoria?

En algunas ocasiones. En los ordenadores pequeños sólo hay espacio para un programa y algunos datos. Pero, en los grandes, la memoria suele estar dividida en secciones, como si fuera un aparcamiento de coches, y cada programa ocupa uno de los espacios reservados.

Algunos ordenadores permiten dividir el programa en porciones, como si fuera una tarta. Luego tratan de ir metiendo esas porciones poco a poco en la memoria.

Ahora, para comprender cómo se puede dividir así un programa, volvamos por un momento a aquellos símbolos binarios.

El 0 y el 1, ¿no?

Sí. La gente que trabaja en el mundo de los ordenadores llama «bits» a estos dos símbolos. Bit, en inglés, quiere decir «trocito». Un trocito de algo. Y eso es justamente lo que es en este caso: un trocito de información. En realidad, es el fragmento de información más pequeño que puedes tener en un programa o en un conjunto de datos. ¡Pueden ser incluso más pequeños que un solo 0 o un solo 1!

Pero los bits por sí solos no sirven para mucho; tienen que combinarse para representar los números, las letras u otros caracteres especiales, como los puntos y las comas, por ejemplo. Todo lo que ves en una máquina de escribir o en el teclado de un terminal está formado por un diferente grupo de bits.

Incluso tenemos un nombre para esos grupos de bits: se llaman «bytes».

No, bytes... con «y griega», y se pronuncia «baits». En la mayoría de los ordenadores, cada byte está compuesto por *ocho* bits. Cada letra del alfabeto, por ejemplo, ocupa exactamente un byte de espacio en la memoria. Esto quiere decir que para formar una «A», o una «X», o una «?» son necesarios ocho bits.

En algunas ocasiones un número determinado es demasiado grande para entrar en un solo byte; entonces unimos varios bytes, y a esta unión le damos el nombre de *palabra*. Una palabra suele tener uno o dos bytes, pero a veces pueden ser más.

Vale. Entonces si unes bits para formar bytes, y, a su vez, los bytes vuelves a unirlos para formar palabras, ¿qué pasa si unes palabras?

Pues lo mismo que cuando tú y yo ponemos varias palabras juntas para formar una frase. En el ordenador las palabras se unen para hacer un *registro*.

En tu colegio seguro que tienen un registro de cada uno de los alumnos, en el que está registrado su nombre, dirección, edad y número de teléfono. Un registro podría ser también un verso de un poema o una instrucción del programa almacenado en la memoria.

Y supongo que también se pueden unir registros.

Claro; del mismo modo que en tu colegio guardan juntas las fichas de todos los alumnos, y los versos de un poema van todos juntos, y las instrucciones de un programa se almacenan todas juntas en la memoria. Un grupo de registros recibe el nombre de *archivo*.

¿Quieres decir algo parecido a un fichero?

Sí, eso es. Hay archivos de todos los tamaños. En realidad, podemos imaginarnos la memoria como un tipo de fichero que contiene las fichas que utiliza la CPU.

Pero recuerda que la memoria no es muy grande; por tanto, el fichero tampoco ha de serlo. ¿Dónde te imaginas que se guardan la mayoría de las fichas?

En los discos y cintas, porque ésos sí que tienen muchísimo espacio.

Correcto. Y la CPU sólo lleva a su memoria el archivo de un disco o de una cinta cuando tiene que utilizarlo inmediatamente.

¿Y después vuelve a dejarlo en su sitio?

No tiene necesidad de hacerlo. Sólo *copia* el archivo en su memoria, del mismo modo que tú copiarías una página de un libro en la pizarra. Ese archivo se *borra* de la memoria cuando la CPU trae uno nuevo, como cuando tú borras la pizarra con el fin de hacer espacio para la siguiente página del libro.

Veamos ahora qué es lo que recuerdas de todo lo que hemos estado hablando. El fragmento más pequeño de información que se puede encontrar en la memoria de un ordenador se llama...

Bit.

Y ocho bits juntos forman un...

Byte.

Y un grupo de bytes se llama...

Palabra.

Y las palabras se pueden unir para formar...

Registros.

Y cuando pones varios registros juntos forman un...

Muy bien. Ahora ya podemos ver cómo la CPU divide ciertos archivos —o programas— en trozos cuando son demasiado grandes para caber enteros en la memoria. Si el archivo contiene en total mil registros por ejemplo, la CPU usará solamente unos 200, y luego sustituirá esos 200 por los siguientes 200, y así, sucesivamente, hasta que llega al final.

¿No hay nada que se quede en la memoria todo el tiempo?

Algunos programas especiales permanecen siempre en la memoria. Tiene que ser así; del mismo modo que tu cerebro siempre recuerda ciertas cosas, como tu nombre, dónde vives o cómo has de hacer para andar, hablar o leer. De no ser así, te verías en apuros, ¿verdad?

¿Esos programas tienen que ocupar un lugar especial en la memoria a fin de que no se borren con los demás?

A veces. Pero normalmente se ponen directamente en una memoria separada.

¿Quieres decir que los ordenadores pueden tener más de una memoria?

Más de un *tipo* de memoria, como le pasa a tu cerebro. El cerebro tiene una memoria a largo plazo que recuerda esas cosas que hemos mencionado, como tu nombre y el lugar en donde vives. Pero también tiene una memoria a corto plazo con la que recuerda cosas tan sólo unos minutos, e incluso segundos, como los nombres nuevos o los números de teléfono, esas cosas que tenemos que apuntar antes de que se nos olviden.

En los ordenadores esa memoria a corto plazo se llama RAM, y es de ésta de la que hemos estado hablando todo este rato. En ella unos programas reemplazan a los otros cuando ya se han utilizado, salvo un programa especial que permanece en ella todo el tiempo. Este programa se llama *monitor*, y tiene un «aparcamiento» especial para él en la memoria.

En los campamentos de verano también tenemos monitores que se encargan de controlar cada una de nuestras actividades.

Eso es más o menos lo que hacen los monitores de los ordenadores. Se ocupan de que todos los programas sean enviados al lugar adecuado de la memoria y de que a todos ellos les llegue la vez para ser ejecutados.

La memoria a largo plazo de los ordenadores se llama ROM. Todos los programas almacenados en la memoria ROM se quedan en ella para siempre y no pueden ser reemplazados por ningún otro. El monitor de los ordenadores pequeños está en la memoria ROM.

¿Y por qué no el de los grandes?

Porque los ordenadores grandes suelen necesitar más de un monitor. Pero como nada más se pueden usar de uno en uno, te tiene que quedar la posibilidad de poder sacar el monitor de la memoria cuando quieras, y eso no se puede hacer con la memoria ROM.

¿Por qué algunos ordenadores necesitan más de un monitor? ¿No hacen todos las mismas cosas?

Así es en los ordenadores pequeños, en los que no hay más de una persona a la vez utilizándolos; la única tarea del moni-

tor en ese caso es introducir en la memoria un solo programa cada vez.

Pero en el caso de los ordenadores más grandes y potentes, se espera que controlen muchos tipos diferentes de tareas; por tanto, necesitan varios monitores, para que cada uno de ellos se encargue de una por separado.

Por ejemplo, el ordenador de una planta química controla la temperatura de los depósitos, y su monitor se llama monitor

de *tiempo real;* pero no es que controle la temperatura solamente una vez, sino que lo hace de un modo continuado y, además, tiene que responder con unas órdenes inmediatas cuando es necesario corregirla.

Por otro lado, el ordenador del hospital lo más seguro es que utilice un monitor a *tiempo compartido*. En este tipo de sistema, el ordenador ya tiene la mayoría de la información almacenada en un disco. El monitor permite que sean muchos los usuarios que utilicen, por turno, esa información desde los diferentes terminales. Lo que sucede es que todo pasa tan deprisa, que da la impresión de que todos están utilizando la información al mismo tiempo.

¿El monitor es un programa grande?

Puede serlo. En los ordenadores grandes, el monitor puede llegar a ocupar un gran trozo de la memoria RAM.

En los pequeños, del tipo de los domésticos, el monitor también es más pequeño y está localizado en la ROM, de modo que no utiliza ningún espacio de la memoria RAM, quedando así toda ella para el programa y los datos.

Pero ahora creo que ha llegado el momento de hacer otro repasito rápido de todo lo que llevamos visto. Vamos a ver... ¿Puedes decirme el nombre de cada una de las diferentes partes del ordenador?

Pues... está la CPU...

¿Y qué hace la CPU?

Se encarga de la aritmética y la lógica... ¡Ah!, y del control.

Correcto. ¿Y qué más?

Después viene la memoria... que puede ser de dos clases.

¿Cuáles son?

ROM, en la que las cosas que se almacenan en ella se quedan allí para siempre y no se pueden cambiar, y RAM, que es donde se ponen los programas y los

datos mientras se utilizan y después se borran para que puedan entrar nuevos datos y programas.

¿Y qué más?

Pues los puertos de entrada y salida, en los que la información entra mediante los mecanismos de entrada y sale mediante los de salida.

¡Muy bien! Otro diez.

Es fácil recordarlas, porque cada parte está en un lugar diferente dentro del ordenador, y no todas juntas, como sucede en el cerebro.

Quizá algún día todas esas partes del ordenador lleguen a formar un todo compacto. Pero cada parte sigue teniendo, por el momento, su propio *chip* independiente.

Un *chip* es una plaquita muy fina de un material especial que puede conducir la electricidad. Tiene el tamaño de un sello más o menos, y su corriente eléctrica puede encenderse y apagarse muy rápidamente.

¡Como los interruptores que almacenaban los números binarios en la memoria!

Sí, esos interruptores utilizan el mismo tipo de material. Tanto la CPU como los puertos de entrada y salida están construidos en *chips,* porque todos ellos procesan información binaria.

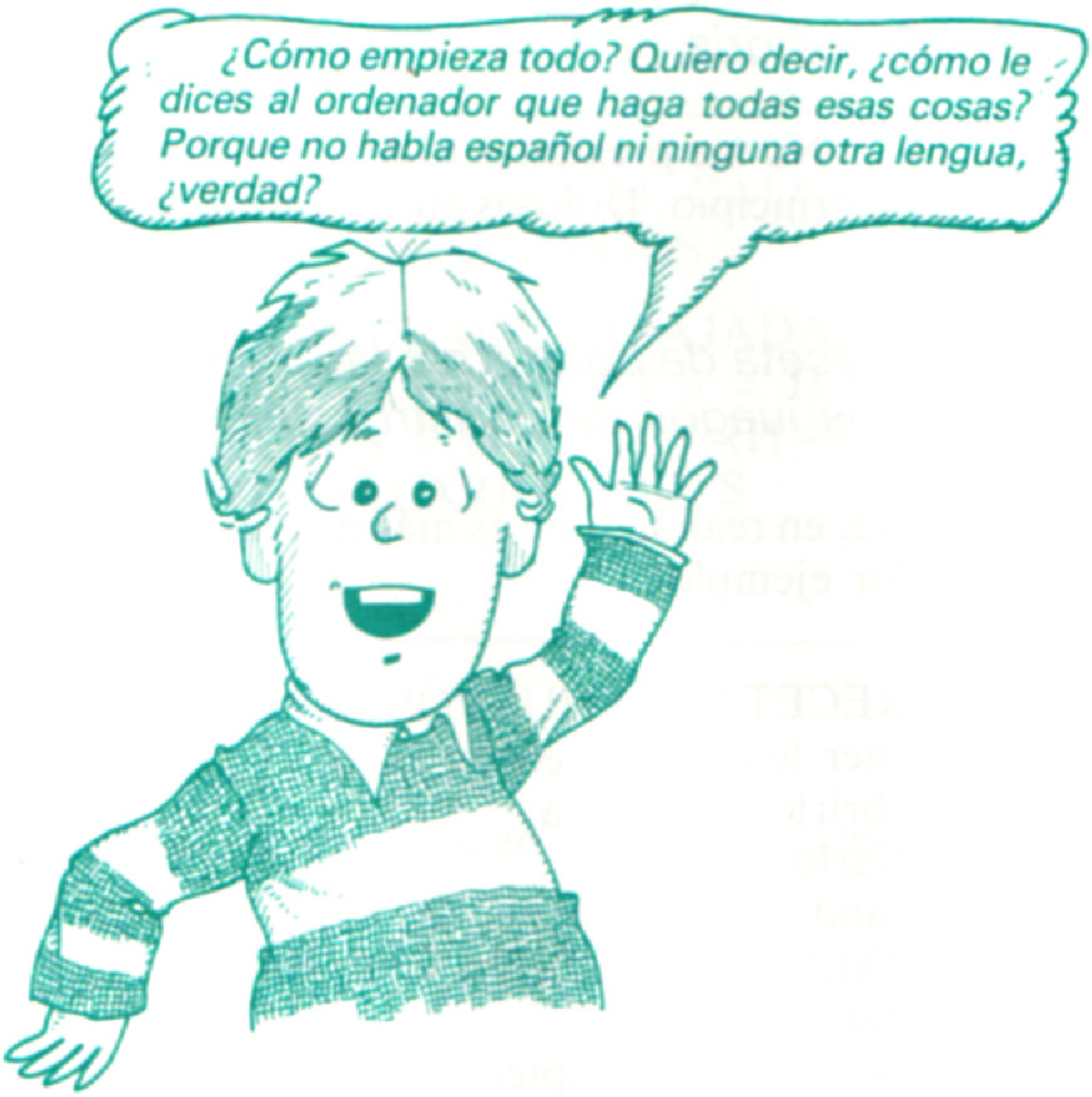
En los ordenadores más pequeños y más sencillos, de hecho, puede estar todo incluido en un solo *chip* pequeñito. Son los que se usan principalmente en cosas tales como los relojes o las calculadoras de bolsillo.

Pero la mayoría de los ordenadores tienen *chips* diferentes para la CPU, la memoria y los puertos de entrada y salida. Están conectados unos con otros sobre una especie de plancha lisa del tamaño de un sobre.

En los ordenadores más grandes, la misma plancha puede contener varios *chips* para la CPU y muchos para la memoria, e incluso puede haber ordenadores que tengan muchas planchas con muchísimos *chips* cada una de ellas. Cuanto más grande y potente sea el ordenador, más *chips* necesitará.

Vale. Entonces tenemos todos esos chips *para almacenar la información y algunos más para hacer el procesado, pero hay una cosa que sigo sin entender.*

¿Cuál?



Algunos hablan inglés, y quizá algún día todos hablen inglés, español y otras lenguas. Pero, por ahora, tenemos que hacer las cosas de una manera un poco diferente. ¿Te acuerdas de lo que hacen los programas?

Sí. Tú dijiste antes que la CPU lo utiliza para decirle a la sección de aritmética qué números tiene que sumar, y a la sección de lógica qué decisiones tiene que

tomar. Y está almacenado en la memoria junto con los datos.

Eso es.

¿Pero de dónde viene el programa, y quién lo introduce en la memoria, y cómo es, y...?

¡Para, para! No me preguntes todo al mismo tiempo. Empecemos por el principio. Dijimos antes que el programa se parecía a una...

...A una receta de cocina o a las instrucciones que te dan con los juegos de construcciones.

Eso; porque, en realidad, no es más que un conjunto de instrucciones. Por ejemplo:

RECETA DE HUEVOS COCIDOS

1. Poner los huevos en un cazo.
2. Cubrirlos con agua.
3. Dejarlos hervir.
4. Cuando rompan a hervir, bajar el fuego.
5. Dejarlos cocer a fuego lento durante 10-12 minutos.
6. Pasarlos a un recipiente con agua fría.

INSTRUCCIONES PARA MONTAR UN CARRITO DE JUGUETE

1. Fijar las abrazaderas del soporte de los ejes en las esquinas inferiores de la estructura.
2. Pasar los ejes por los agujeros de las abrazaderas.

3. Poner las ruedas en los ejes.
4. Asegurarlas con sus tapaderas correspondientes.
5. Fijar la abrazadera del tirador a la tabla.
6. Unir el tirador a la abrazadera utilizando un tornillo y una tuerca.
7. Pelar las calcomanías y pegarlas en el cuerpo del carrito.

PROGRAMA PARA EL CALCULO
DE SALARIOS

```
10  INPUT "HORAS TRABAJADAS"; H
20  IF H < = 40 THEN LET S = H * 10
    ELSE LET S = (40 * 10) + (H — 40) * 15
30  PRINT "SALARIO: Pts"; S
40  END
```

Ahora dime, ¿qué tienen en común todos estos ejemplos?

Bueno... cada paso es muy pequeño.

Sí. ¿Y qué más?

Todos ellos están numerados, para que los hagamos por orden.

Eso es muy importante. ¿Algo más?

No se me ocurre nada más.

¿Se pueden usar más de una vez?

Bueno, la receta y las instrucciones sí, así que supongo que el programa también.

Exactamente. Y quizá esto sea lo más importante. Una vez que está escrito y *almacenado* en algún sitio, el programa puede usarse una y otra vez, tantas veces como sea necesario.

Tú lo has dicho. Una vez que está almacenado en un disco o cinta, la CPU puede copiarlo en la memoria en forma de entrada *(input)* y utilizar sus instrucciones cada vez que lo necesite.

Por tanto, podemos decir que un *programa* es un conjunto de instrucciones almacenadas en un disco o una cinta que se copian en la memoria del ordenador para decirle a la CPU qué es lo que tiene que hacer.

Pero sigo sin entender de dónde sale y cómo llega al ordenador.

Bueno, al igual que la receta o las instrucciones para montar el carrito, alguien tuvo que escribirlo primero. Un *programador* es una persona que escribe programas.

¡Eso sí que es fácil de recordar!

Pues sí. El programador decide qué tipo de trabajo tendrá que hacer el ordenador, y luego escribe un conjunto de instrucciones para indicarle paso a paso cómo ha de hacerlo. Como la persona que escribió la receta tuvo que escribir todos los pasos precisos para cocer un huevo.

Pero no puedes escribir las instrucciones así como así. Con los ordenadores tienes que decir exactamente lo que quieres hacer; de lo contrario, el ordenador puede interpretarte mal.

¿Conque eso crees, eh? A ver, comprobemos si es cierto eso que dices. Imagínate que quieres que yo sume dos números, ¿cómo me pedirías que lo hiciera?

Dejáme pensar, déjame pensar... Ya está: «Suma los números 8 y 3, por favor.»

¿Que los sume a qué?

Bueno, uno al otro, por supuesto.

¡Ah!, pero no dijiste eso antes, ¿verdad? Podrías haber querido decirme que los sumara a cualquier otro número, el 5, por ejemplo.

Ya, pero pensé que comprenderías lo que quería decir.

En realidad, sí que lo comprendí; pero un ordenador podría no haberlo comprendido. Bueno, vale. ¿Qué harías después?

Pues tendrías que esperar sentado, porque...

¡No me lo digas! Porque no te dije que me la dieras.

¡Eso es! Solamente me dijiste que sumara los números.

¿Ves? Un ordenador solamente hará lo que le digas que haga, ni más ni menos.

¡Huy! ¿Y cómo se acuerda el programador de todas esas cosas?

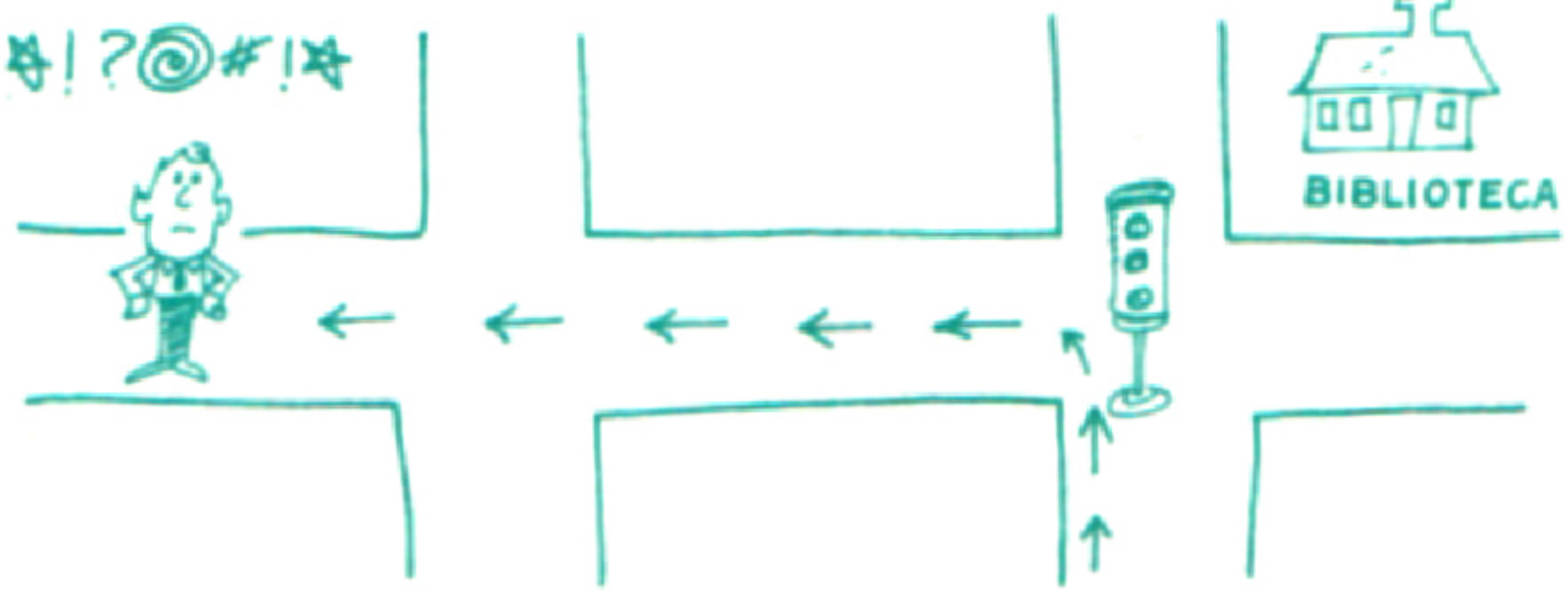
Porque tiene muchísima práctica. Cuando lo has hecho algunas veces es mucho más fácil.

Otra cosa muy importante es que las instrucciones estén exactamente en el orden adecuado. Por ejemplo, ¿qué pasaría si hiciéramos el paso 3 de la receta antes del 2?

¡Menudo lío entonces! ¿Eh?

Asimismo, toda la información ha de ser correcta, incluso los más pequeños detalles.

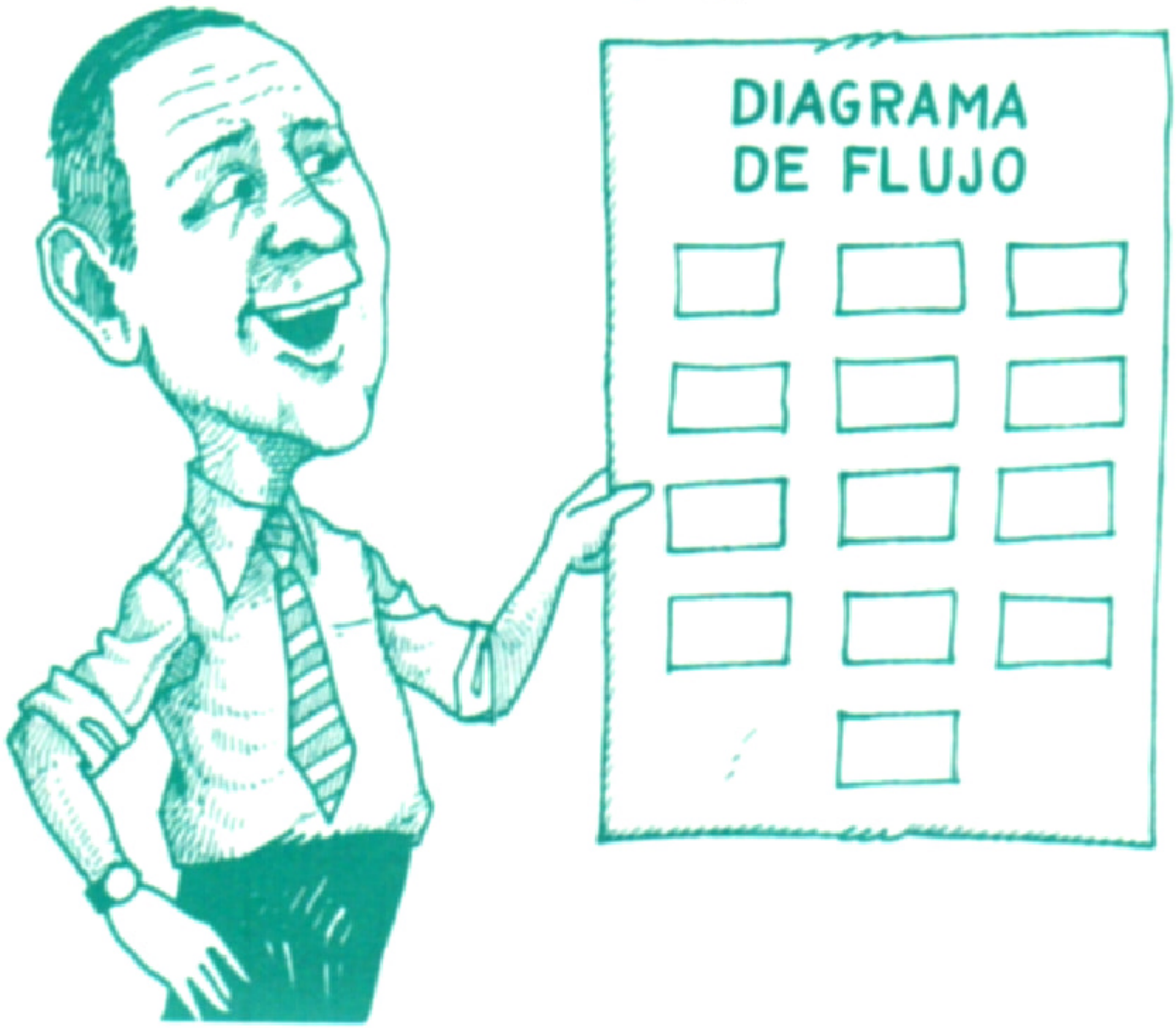
Por ejemplo, si alguien te preguntara cómo se va a la biblioteca, ¿qué pasaría si tú le dijeras que torciera a la izquierda, en vez de a la derecha, al llegar al semáforo? Pues que esa persona se perdería sin remedio.



Así pues, los programas tienen que ser totalmente correctos. Un solo error, y la policía puede arrestar a un hombre inocente. O un satélite puede estallar durante su lanzamiento. O un depósito de productos químicos puede inundar toda una fábrica.

O unos grandes almacenes podrían cobrarte de más por algo que has comprado y pagado con tarjeta.

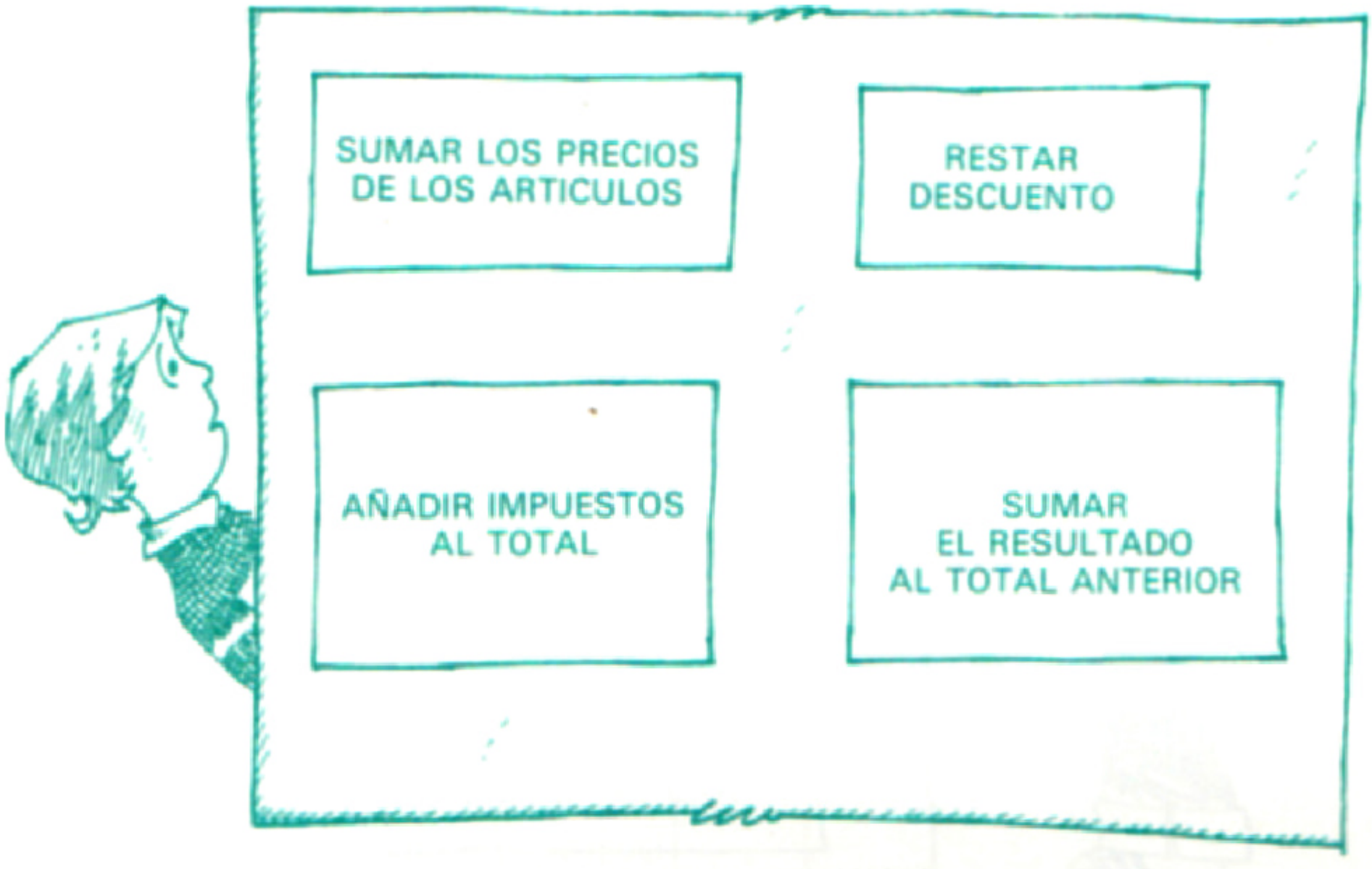
También. En vista de todas estas cosas, el programador generalmente planifica con todo cuidado su programa, a fin de asegurarse de que todos los pasos son correctos y de que están en el orden adecuado. Para ello, suele hacer antes un dibujo de las cosas que ha de hacer el programa.



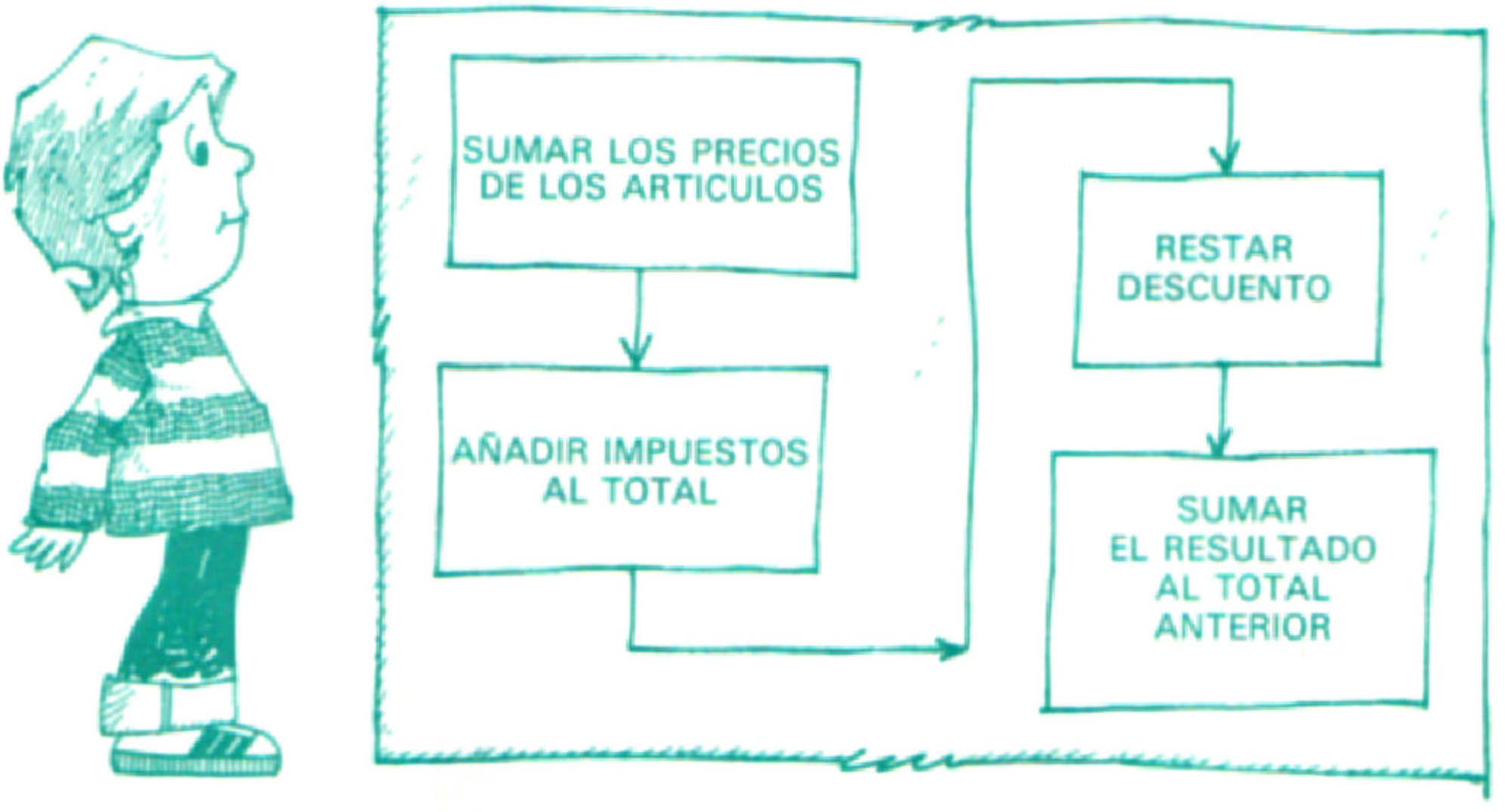
Este dibujo recibe el nombre de *diagrama de flujo*, porque un diagrama es una especie de mapa o gráfico, y flujo significa el movimiento de un sitio a otro; por ejemplo, los ríos fluyen entre una y otra ciudad o pueblo. ¿Te acuerdas de lo que decíamos cuando hablamos de los símbolos?

Sí. Los números están formados por símbolos y letras, y los signos de puntuación son símbolos, porque representan cosas.

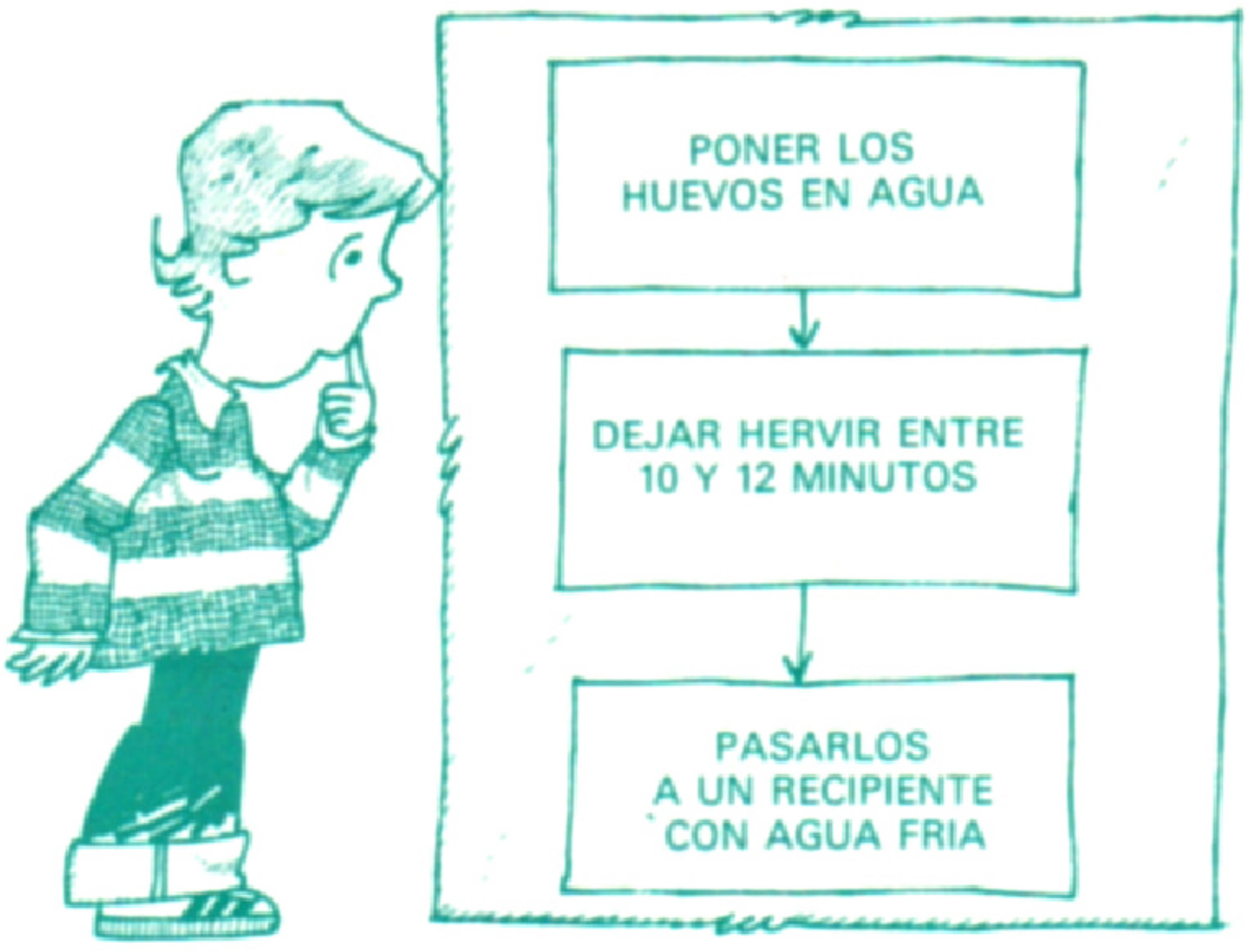
Bueno, pues los programadores también usan símbolos en los diagramas de flujo. Cada paso del programa, por ejemplo, está representado por un rectángulo. Y dentro de éste se escribe lo que se ha de llevar a cabo en ese paso.



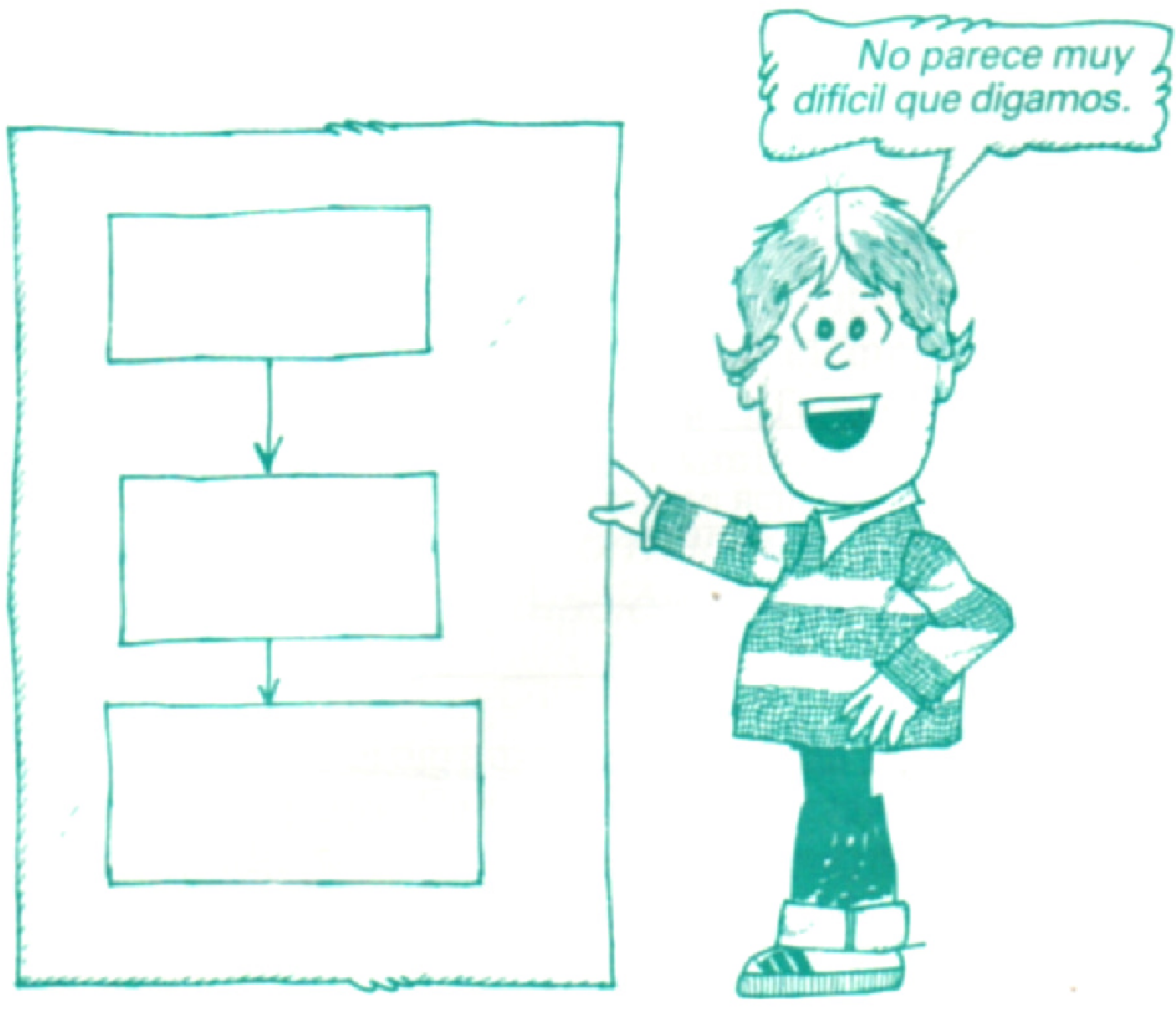
Luego unimos un rectángulo a otro mediante flechas, para mostrar cuál viene primero, segundo, tercero, etc.



Por ejemplo, el diagrama de flujo para una receta de huevos cocidos sería el siguiente:



Observa que en el diagrama de flujo algunos de los pasos se combinan, a fin de no tener que hacer tantos rectángulos.



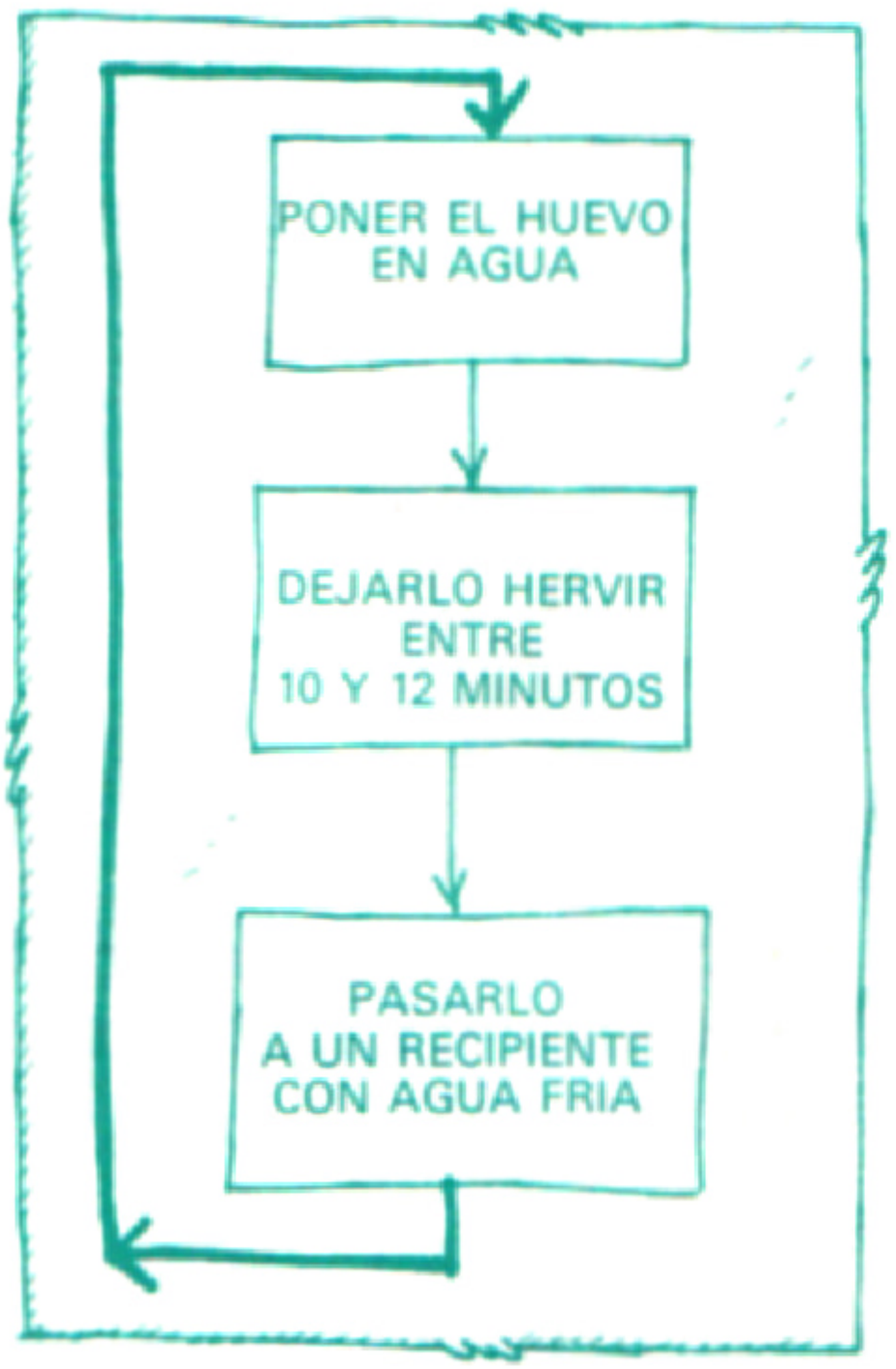
Eso porque es una receta sencilla. Pero cambiemos un poco. Imagínate que solamente tuviéramos un recipiente pequeño, en el que sólo cupiera un huevo.

En ese caso tendríamos que seguir la receta más de una vez.

Por tanto, añadiríamos otra instrucción que dijera:

7. Repetir los pasos mencionados más arriba, para hervir el resto de los huevos.

Pero en el diagrama de flujo podemos dar esta instrucción sencillamente añadiendo otra flecha:



En el lenguaje de los ordenadores este procedimiento recibe el nombre de *bucle*.

Porque se repite una y otra vez hasta que ya no quedan huevos por hervir.

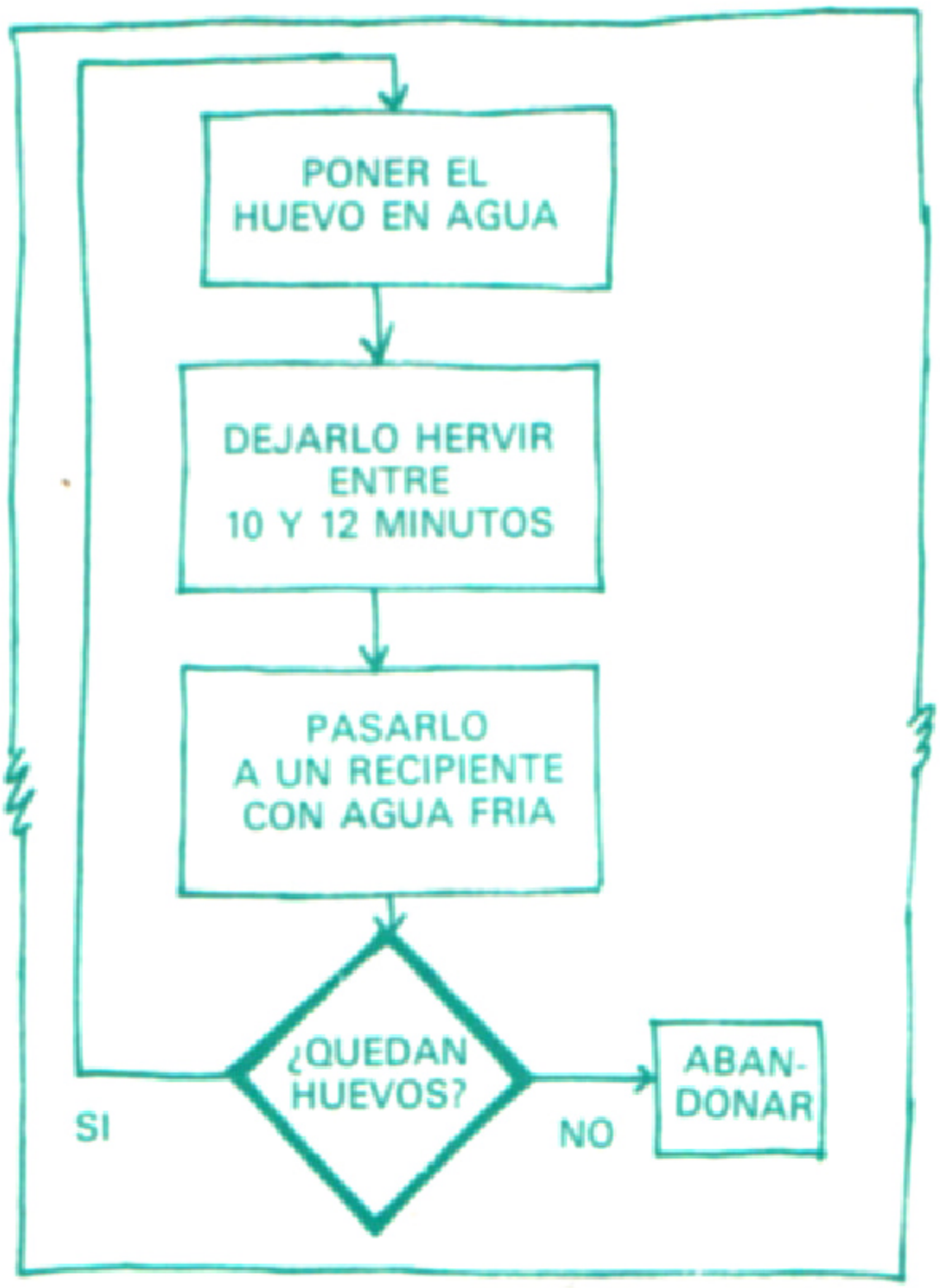
Eso es. Independientemente de que tengas mil huevos o solamente uno, una vez que el programa ha sido introducido en la memoria se pueden repetir las mismas instrucciones cuantas veces se quiera.

¡Estupenda observación! Y si queremos que nuestro diagrama sea absolutamente preciso, deberíamos hacer algo para solucionar ese problema. ¿Qué sugieres tú?

¿Añadir otro rectángulo?

No exactamente. Para este tipo de casos tenemos otro símbolo diferente.

Cada vez que lleguemos al final de la receta, haremos la siguiente pregunta: ¿Quedan huevos? Si la respuesta es SI, una flecha nos vuelve a llevar al principio del programa; si la respuesta es NO, lo abandonamos. Así que nuestro nuevo símbolo tendrá este aspecto:



Este símbolo muestra que estamos tomando decisiones. ¿Te recuerda eso algo?

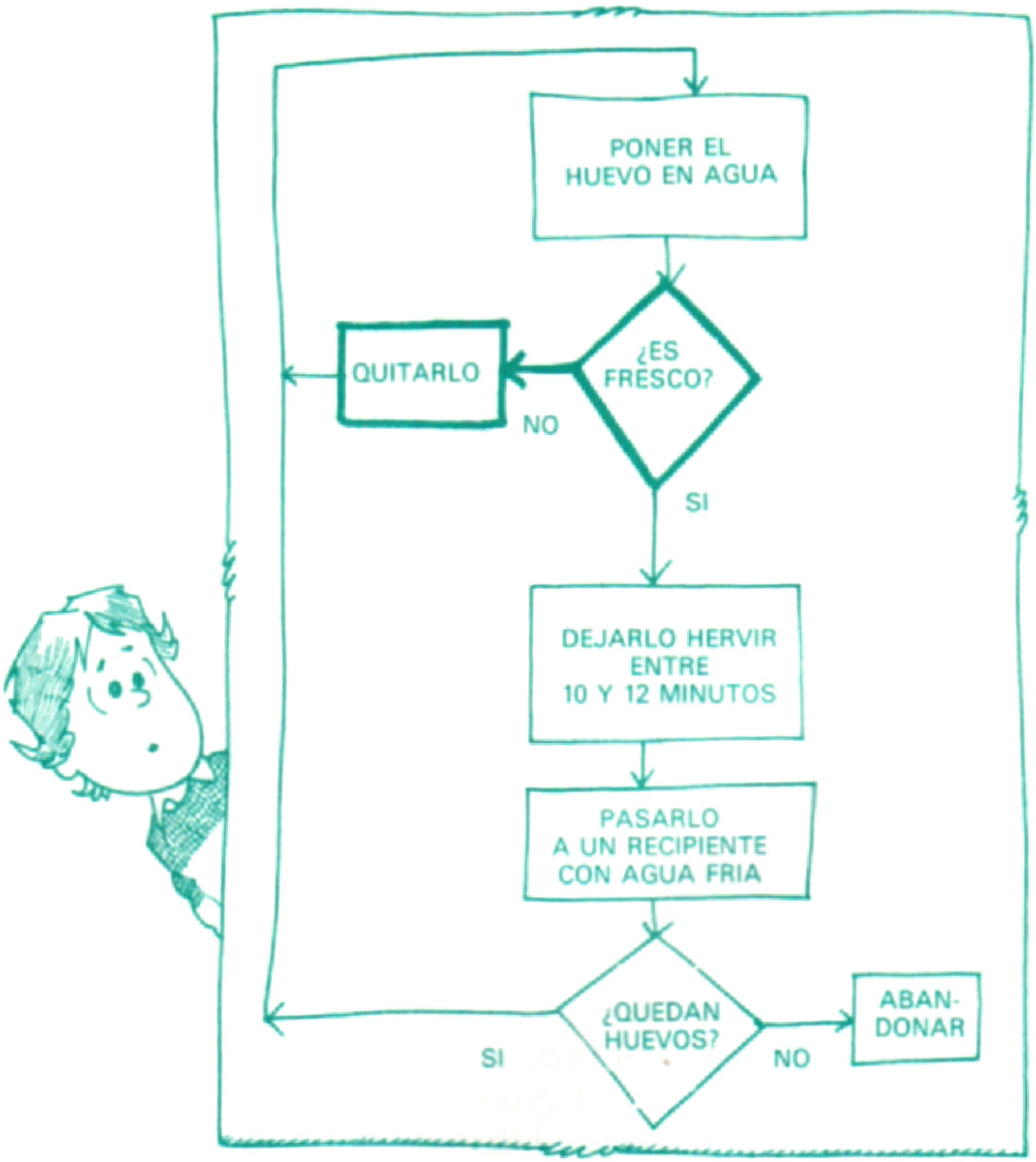
Pues claro, a la sección de lógica de la CPU, en la que un 1 es SI y un 0 es NO.

Eres un chico listo. La toma de decisiones es una parte muy importante del trabajo del ordenador. Y cualquier programa típico incluirá cantidad de decisiones.

Añadamos otra decisión a nuestra receta. ¿Sabes cómo se reconocen los huevos que no están muy frescos?

Sí. Flotan en el agua en lugar de irse al fondo del cazo.

Eso es. Por tanto, hagamos una comprobación para asegurarnos de que sólo vamos a cocer huevos bien frescos. En el caso de que hubiera huevos poco frescos, habría que sustituirlos:



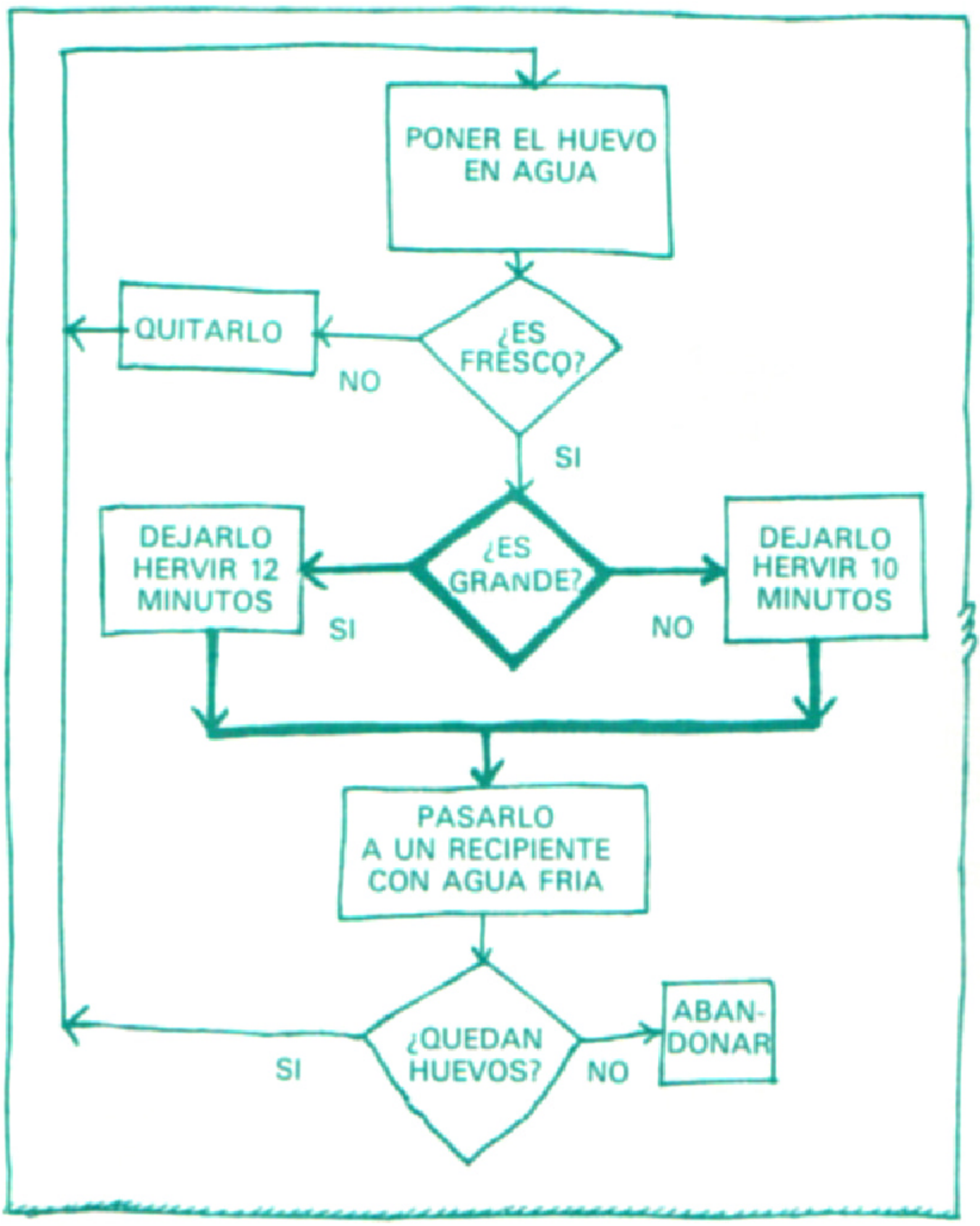
Esta vez el símbolo que representa la toma de decisiones nos permite saltarnos el resto del programa. La flecha que sale del NO vuelve a llevarnos al inicio para probar con un huevo

nuevo, mientras que la que sale del SI nos permite continuar como lo hacíamos antes.

Añadamos una más. Imagínate que tenemos huevos grandes y pequeños. Los grandes tardan más en cocerse que los pequeños. ¿Cómo crees que podríamos comprobarlo?

Tendríamos que introducir un símbolo de toma de decisiones antes de la instrucción que especifica el tiempo de cocción.

Eso es. Y quedaría así:

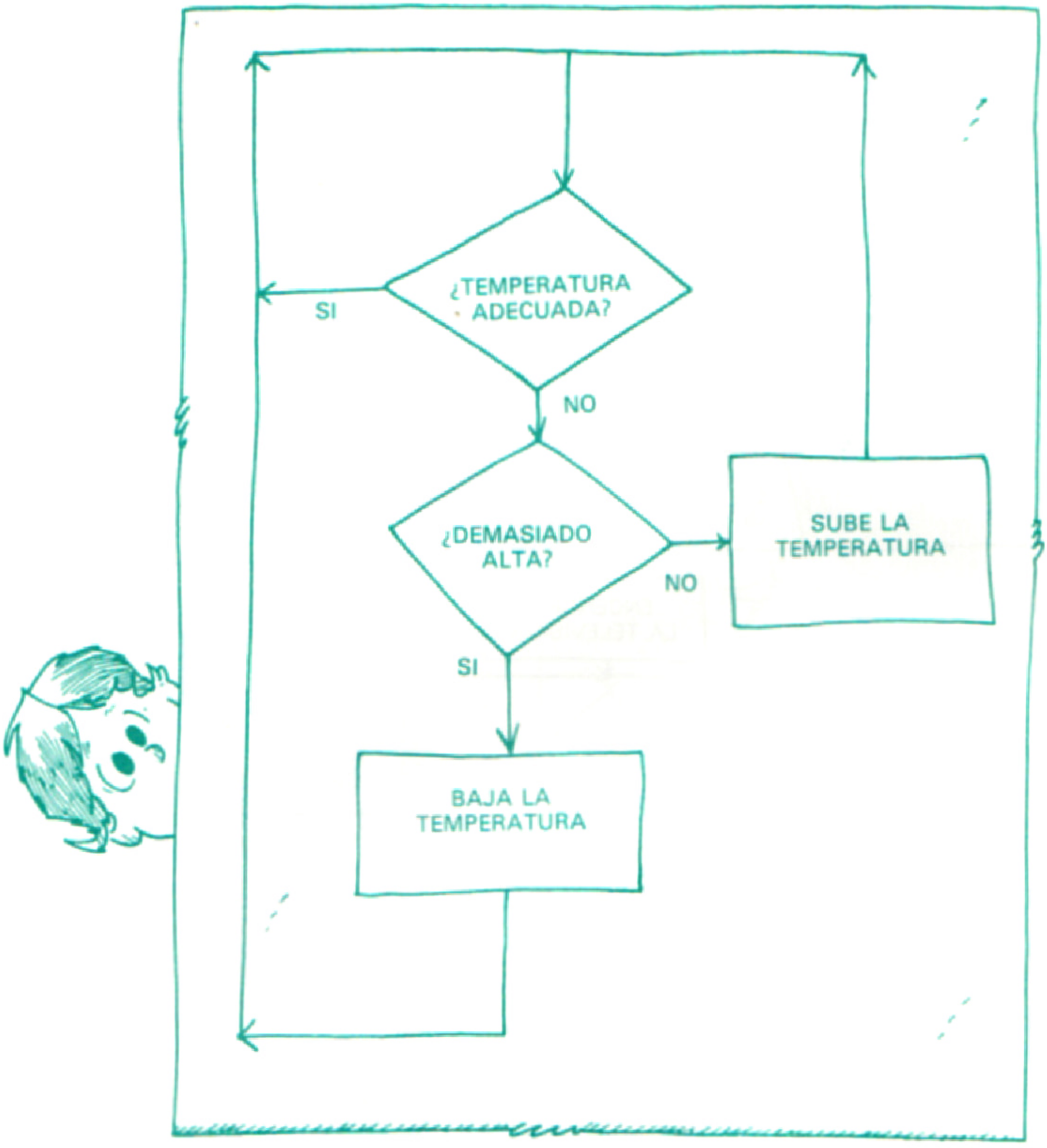


Así que ahora ya hemos visto que los símbolos que representan la toma de decisión nos dicen cuándo tenemos que repetir una instrucción, cuándo la tenemos que saltar y cuándo tenemos que escoger entre dos instrucciones diferentes.

Es verdad, aunque a veces eso es exactamente lo que queremos hacer.

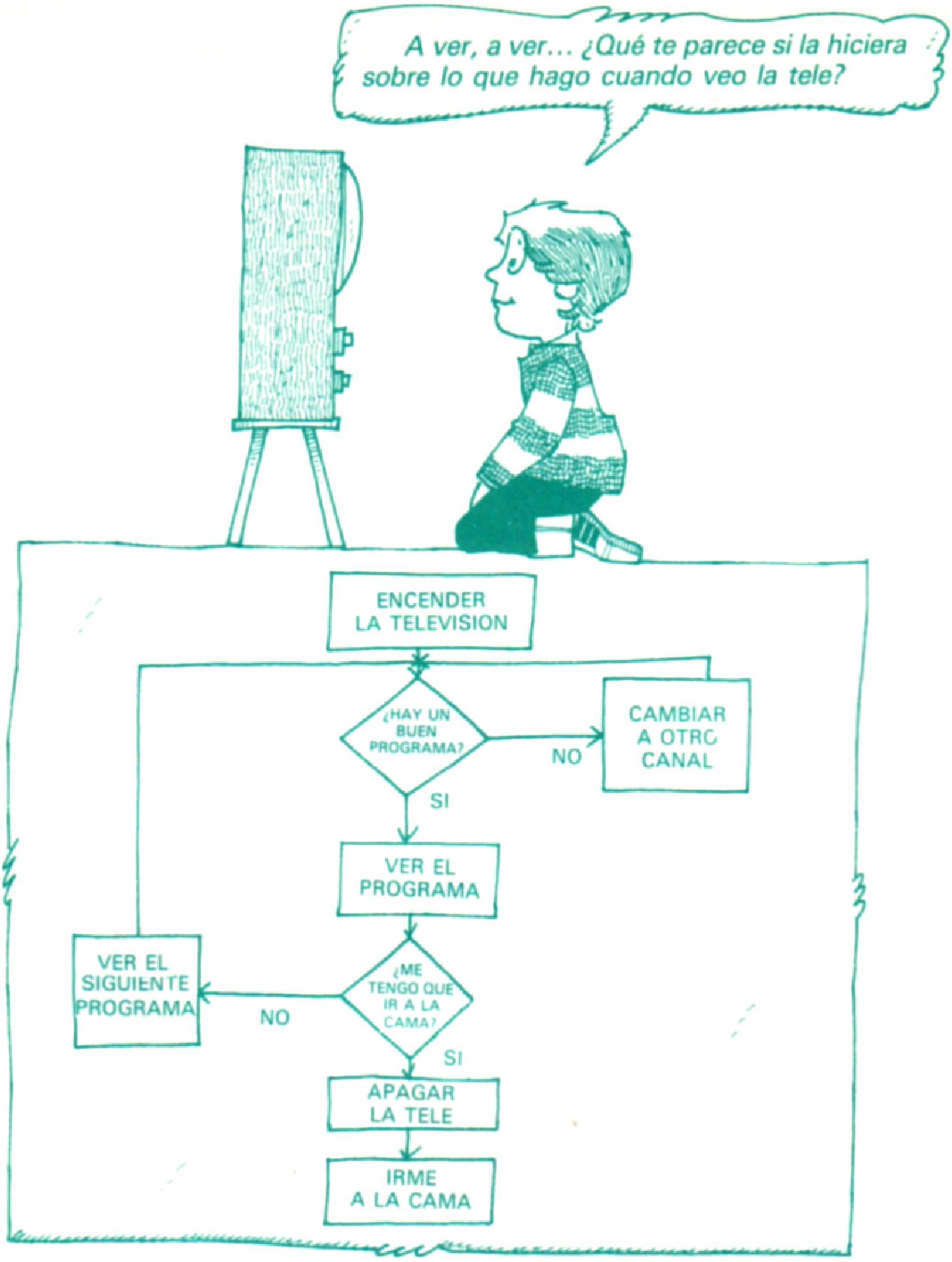
¿Sí?

Claro. Piensa en el ordenador que comprobaba las temperaturas de los depósitos en la planta química. Tiene que continuar comprobándolas aun después de haberlas rectificado:



Esto recibe el nombre de *bucle sin fin,* porque nunca deja de repetir las mismas instrucciones, a no ser, por supuesto, que desconectemos todo el sistema.

Bueno, ahora creo que con lo que hemos visto deberías poder hacer tú solo un diagrama de flujo que describa una actividad cualquiera que se te ocurra. ¿Por qué no lo intentas?



Está bastante bien, pero me parece que te has olvidado de algo.

¿Sí? ¿De qué?

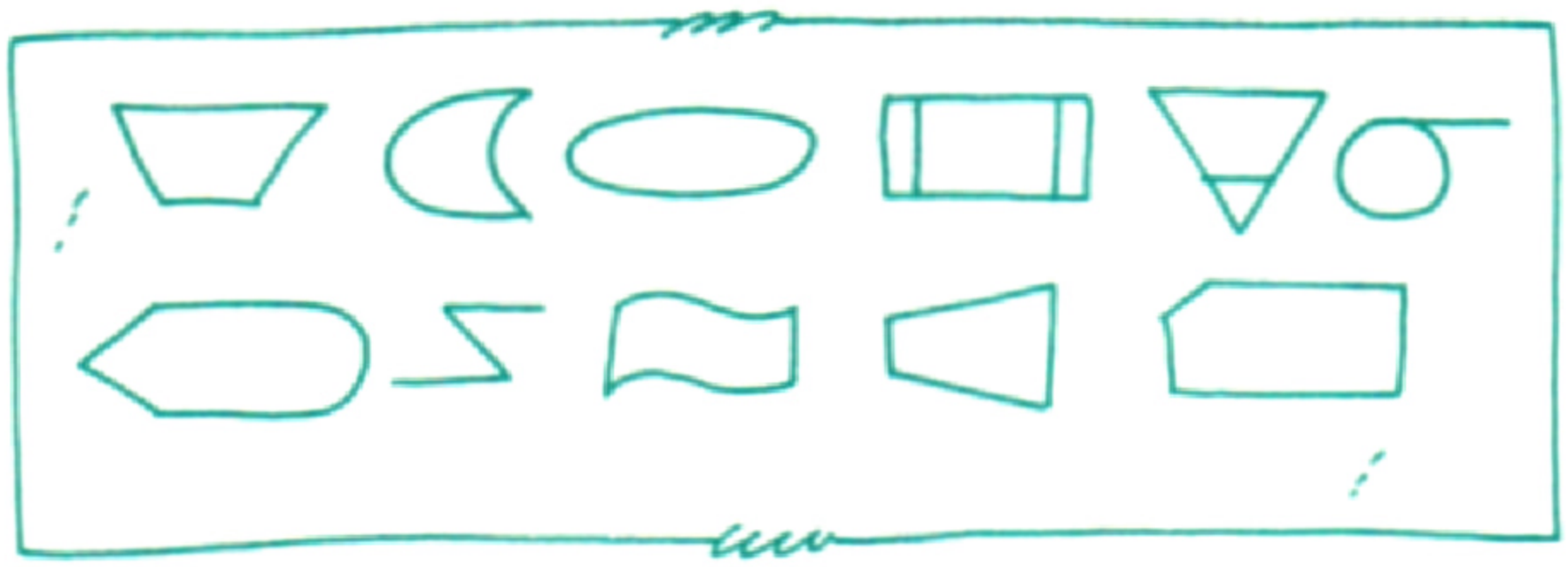
¿Qué pasa cuando estás todo el rato cambiando de un canal al otro y no encuentras un buen programa?

¡Ah, ya veo! El botón no pararía nunca de girar. Eso era un bucle sin fin. ¿Qué te parece así?

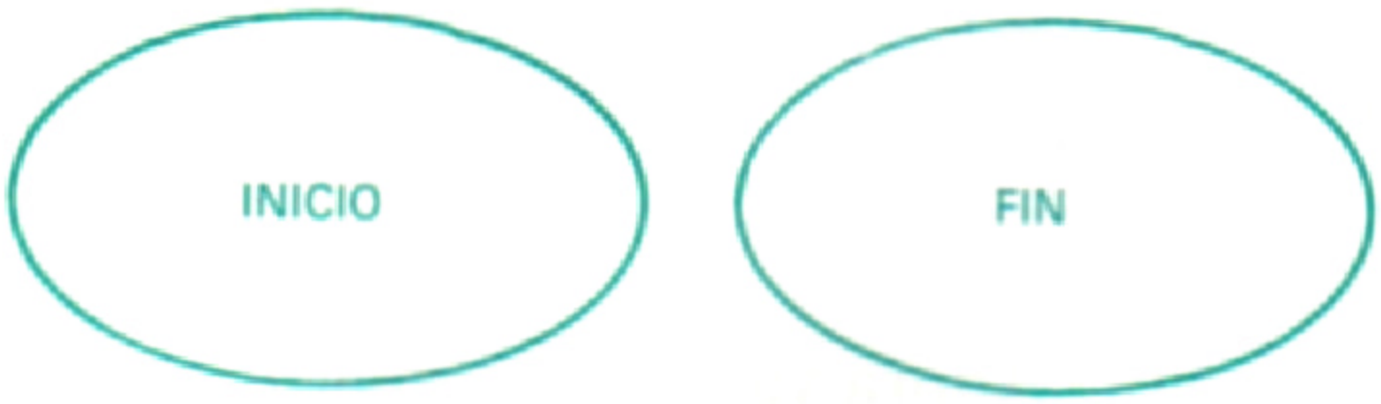
Así está mejor. Ahora ya eres un experto en diagramas de flujo.

¿Hay más símbolos en los diagramas de flujo?

Algunos programadores utilizan muchos símbolos diferentes, como estos:



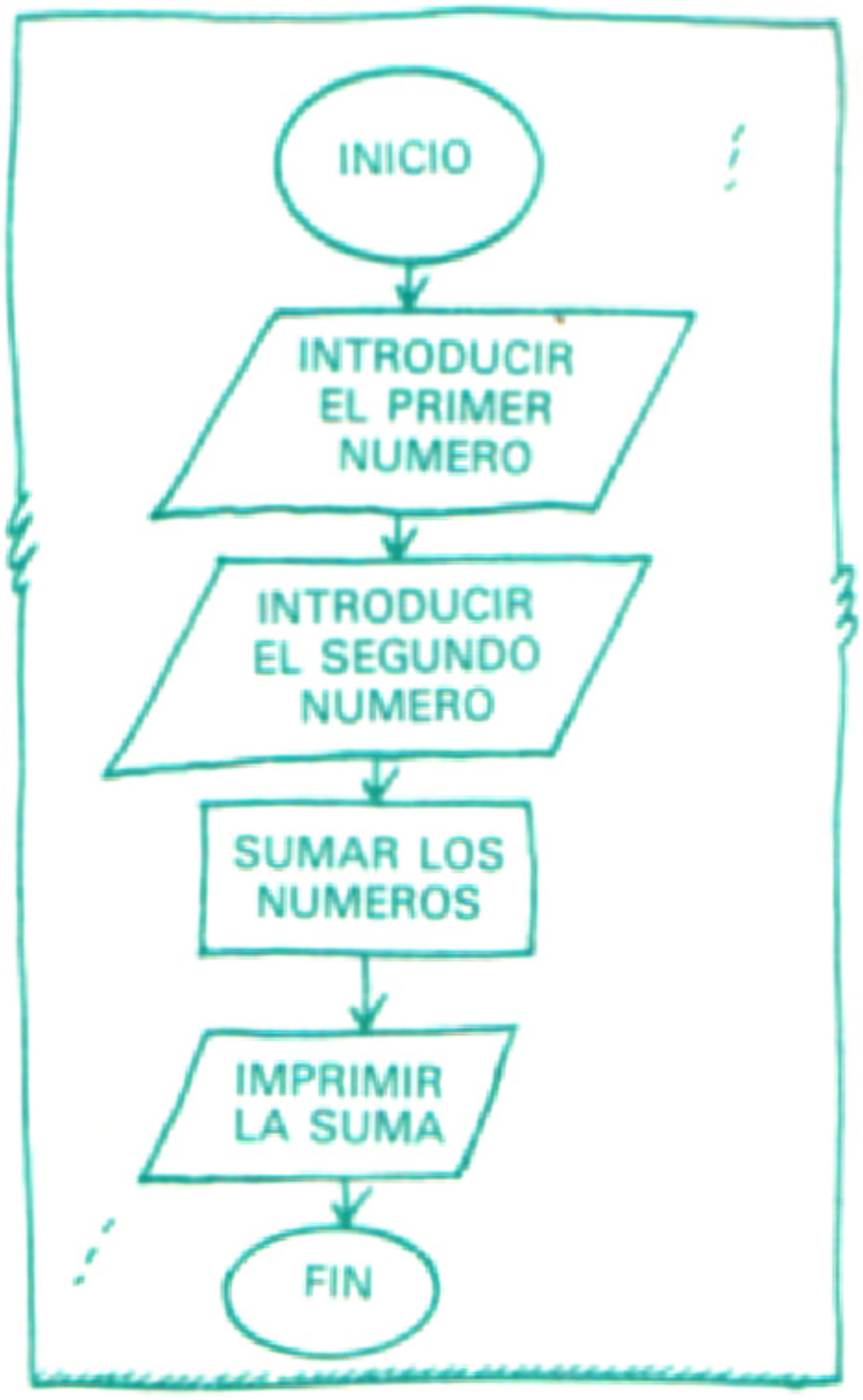
Pero, en la mayoría de los casos, cuantos menos utilices, mejor. Podemos incluir:



que muestran dónde empieza y acaba el diagrama de flujo. Además, podríamos necesitar:

para mostrar el lugar en el que hay que introducir los datos y el lugar donde van las respuestas.

Suponte que te digo que sumes dos números. El diagrama de flujo sería el siguiente:



Parece tirado, ¿verdad? Pero imagínate que te diera un tercer número, ¿qué harías?

Lo añadiría a la suma de los dos primeros.

¿Y si te diera un cuarto?

Lo mismo. Añadiría ese nuevo número a la suma.

¿Y si siguiera dándote más y más números?

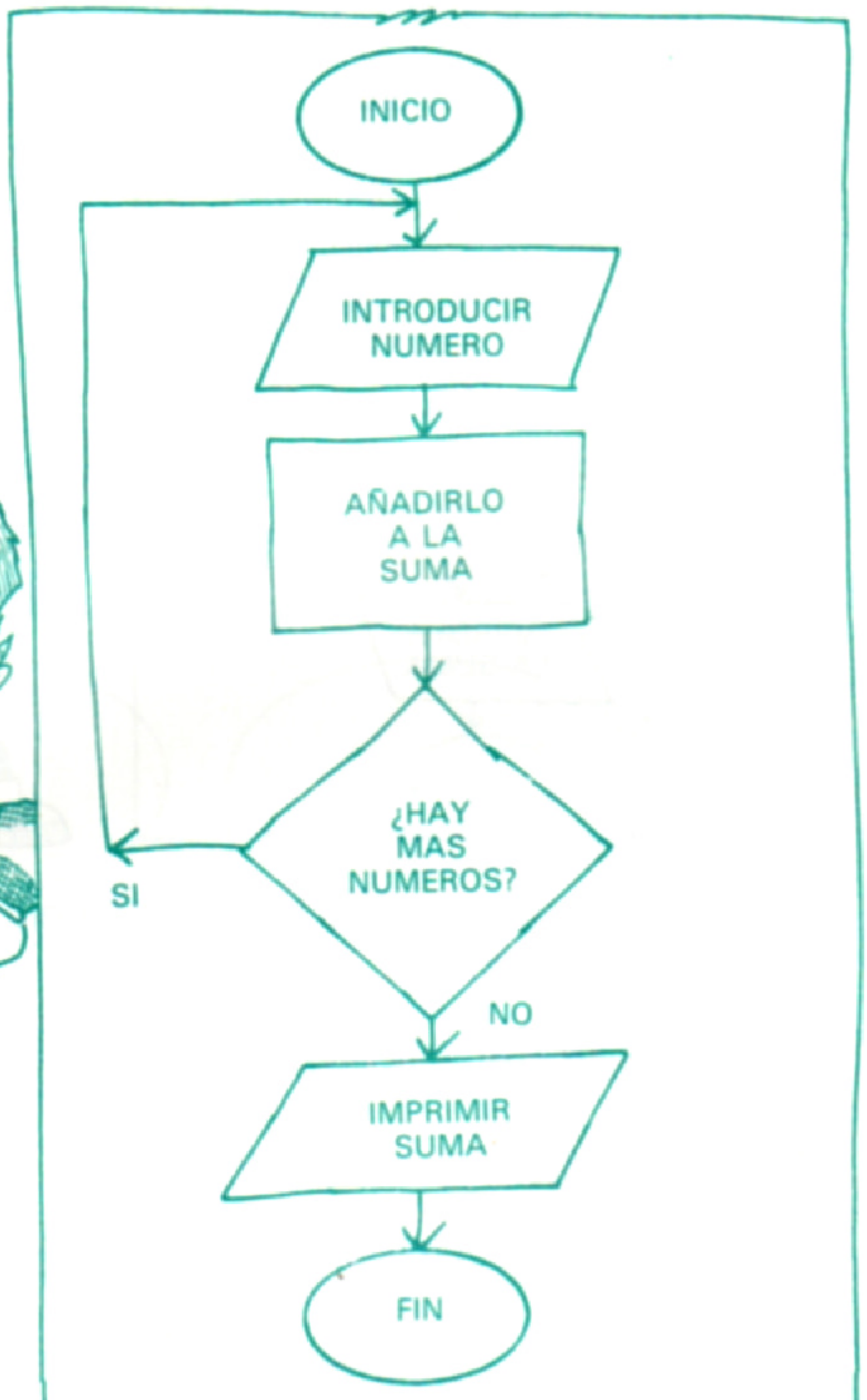
Seguiría añadiéndolos a la suma.

¿Hasta...?

Hasta que dejaras de darme números.

Eso es. ¿Crees que podrías hacer el diagrama de flujo de esta suma?

Voy a intentarlo. ¿Qué te parece éste?:



¡Muy bien! Ahora dime, ¿sabes qué es un promedio? Como la media de edad de los niños y niñas de tu clase.

Sí, claro. Hemos hecho ese tipo de problemas en clase de matemáticas. Primero sumas todas las edades, y después divides el resultado de la suma por el número de niños.

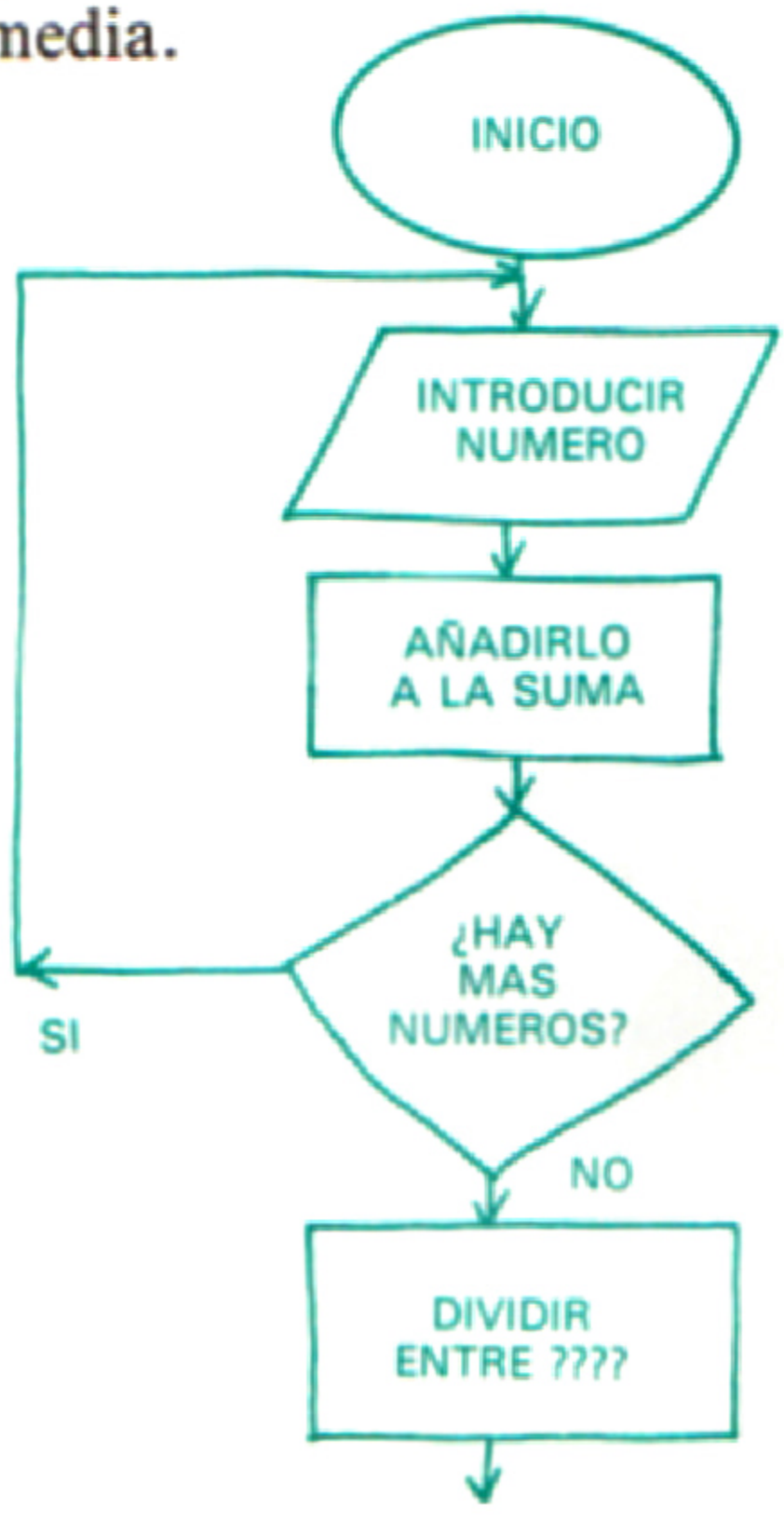
EDAD 9 EDAD 10 EDAD 9 EDAD 12

$$9 + 10 + 9 + 12 = 40$$

$$\frac{40}{4} = 10$$

10 = MEDIA DE EDAD

Vale; pues hagamos como si esos números que estábamos sumando representaran las edades de tus compañeros de clase. Y cuando sepamos el resultado de la suma, lo dividiremos para saber la media.

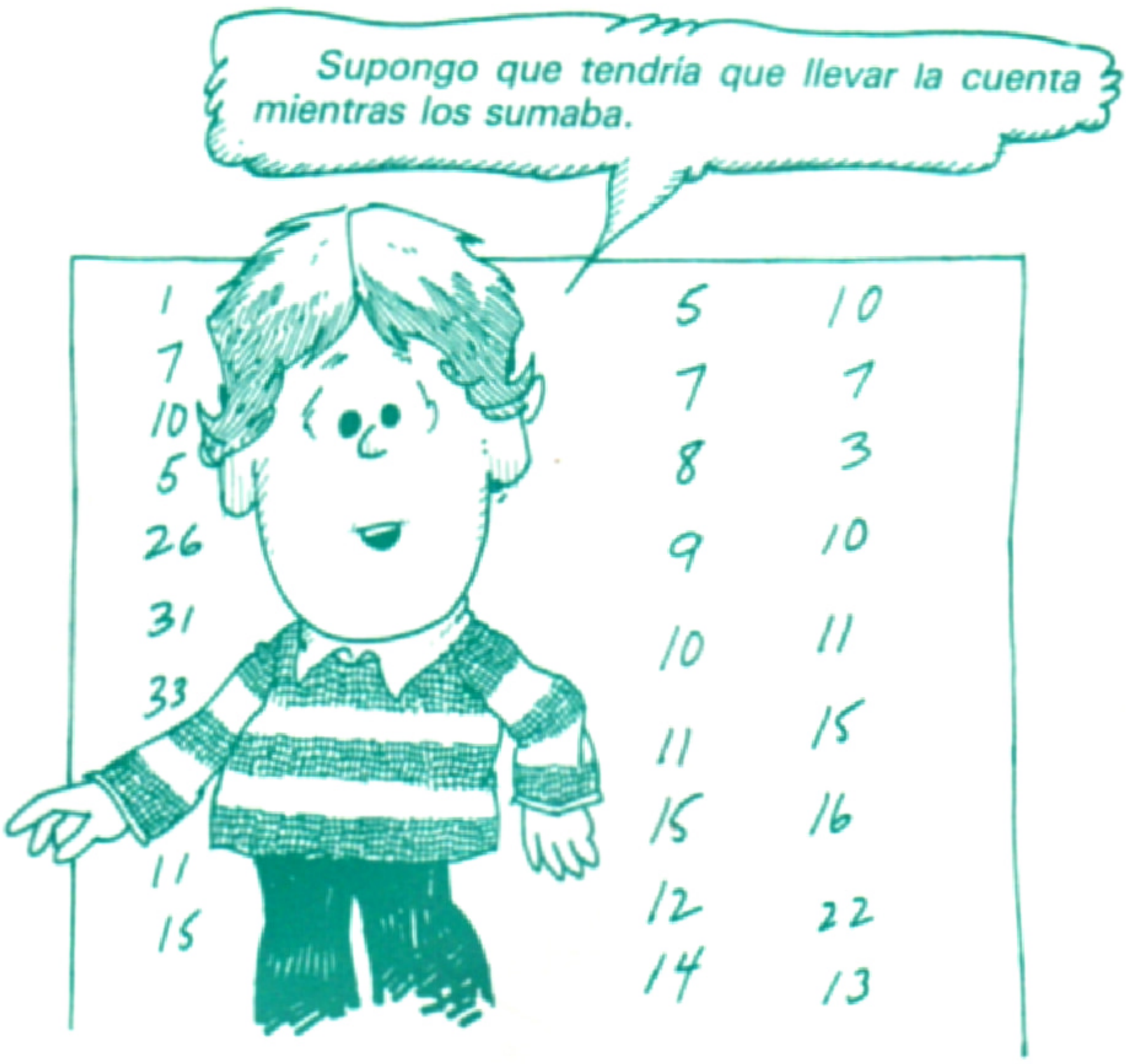


¡No podemos hallar la media, porque el diagrama de flujo no nos dice cuántas edades hemos sumado! Ahora no sabemos por cuánto hemos de dividir.

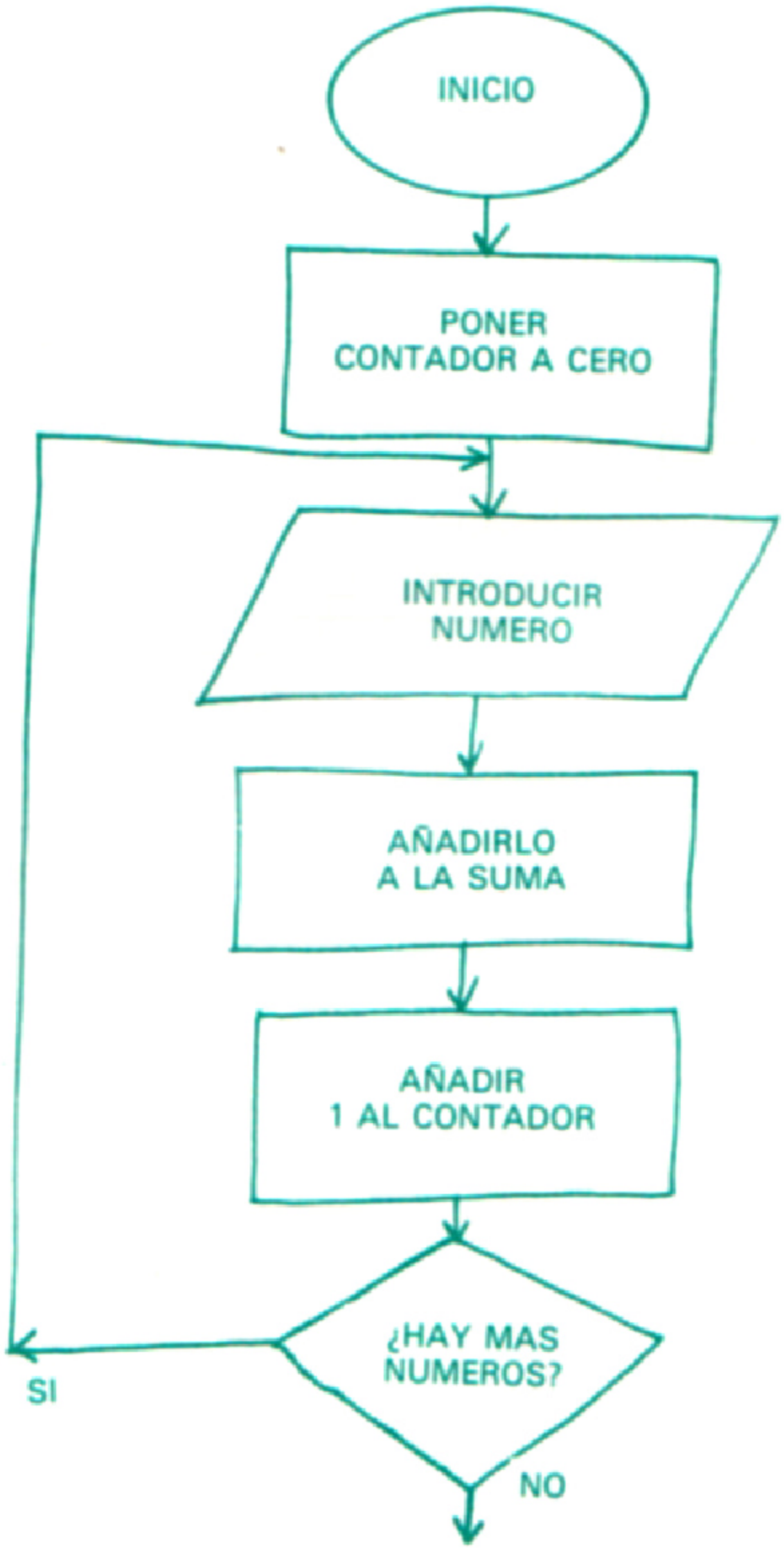
Tienes toda la razón. Y si queremos usar el diagrama de flujo para una clase diferente, es posible que el número de alumnos sea otro distinto.

¿Y qué hacemos entonces?

Bueno, imagínate que estuvieras sumando muchos números y tuvieras que saber exactamente cuántos son: ¿qué harías?



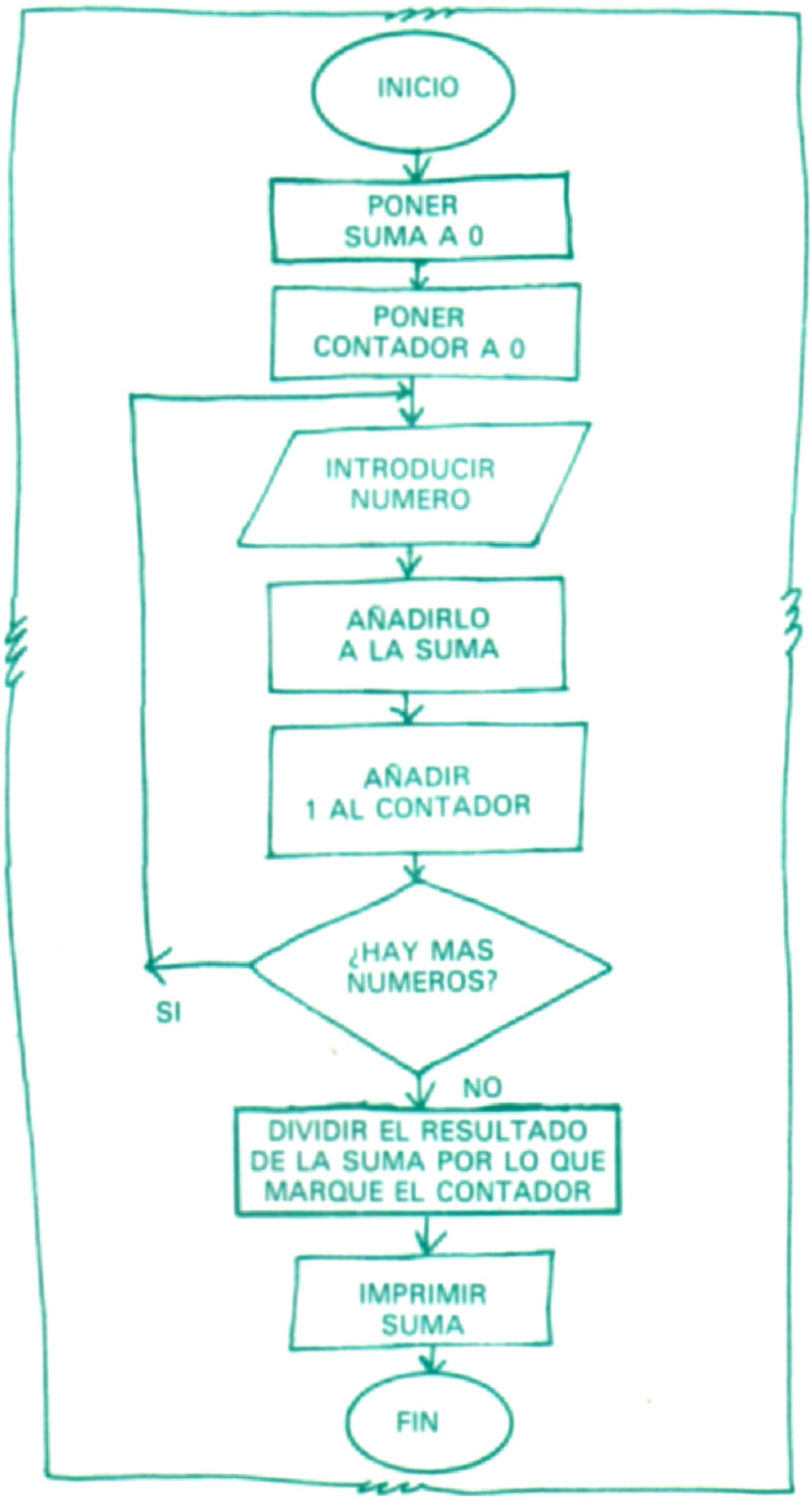
Bueno, pues en nuestro diagrama de flujo deberíamos poder hacer lo mismo. Incluyamos un «contador» y pongámoslo a 0; luego, cada vez que añadamos una nueva edad a la suma, añadiremos también 1 al contador.



Luego, cuando hayamos añadido la edad del último alumno, el contador nos dirá cuántas son las edades que hemos sumado en total.

Y entonces ya podremos dividir el resultado de la suma por lo que marque el contador, y así sabremos la media.

Eso es. Pero hay algo más. Como al empezar tenemos que poner el contador a 0, también tendremos que poner la suma a 0 y entonces nuestro diagrama de flujo tendrá este aspecto:



Ahora empezamos a aproximarnos a lo que sería un verdadero diagrama de flujo realizado por un programador.

Pues claro. Lo mismo que un arquitecto no construiría una casa sin haber dibujado antes los planos.

Y los escritores suelen hacer un esquema del libro antes de empezar a escribirlo.

Y los pintores hacen un boceto del cuadro que quieren pintar.

¿Pero cuándo empieza a escribir el programa el programador?

En cuanto está seguro de que el diagrama está bien. Entonces decide qué lenguaje irá mejor para ese programa en concreto.

¿Lenguaje? ¿Francés o inglés?

No; a lo que me refería era a lenguajes como el FORTRAN o el COBOL.

El inglés y el francés son lenguas que se hablan, además de escribirse. Las estudiamos para poder entendernos con la gente que las habla y leer las cosas que han escrito.

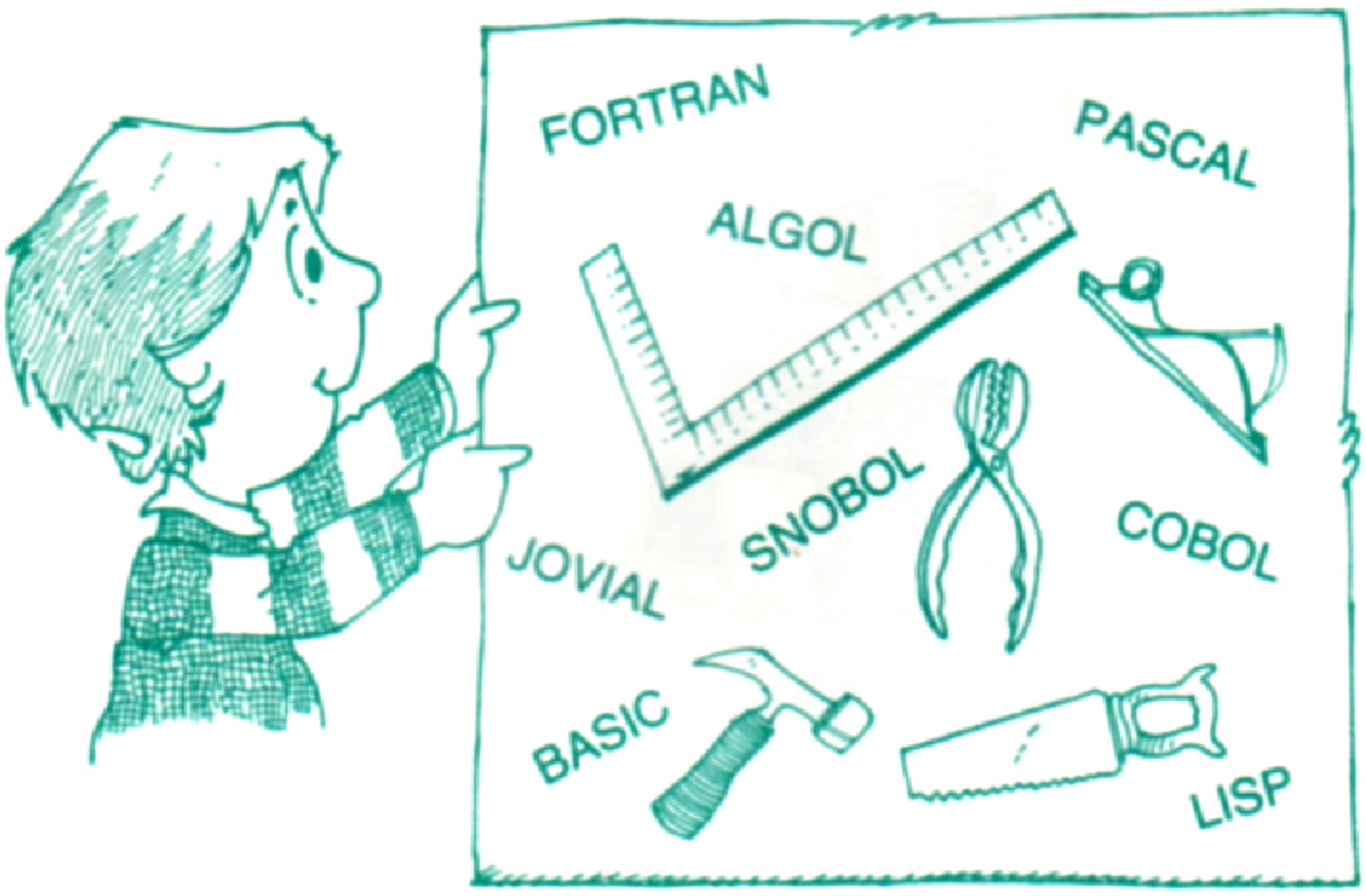
FORTRAN y COBOL son lenguajes informáticos y sólo son escritos. Pero tienen la misma función: nos permiten comunicarnos con el ordenador, ya que éste no habla español ni ninguna otra lengua.

¿Y por qué necesita saber dos lenguas el ordenador?

¡Oh! ¡Hay muchas más de dos! Se aproximan a mil los lenguajes informáticos existentes en la actualidad. Si he hablado del FORTRAN y el COBOL es porque son dos de los más conocidos.

La razón de que haya tantos es que el ordenador tiene que llevar a cabo una gran cantidad de tareas diferentes. Y unos lenguajes son mejores que otros para ciertas tareas.

Del mismo modo que el carpintero utiliza una herramienta u otra dependiendo del trabajo que esté realizando, el programador usa el lenguaje que mejor se adapta al problema que hay que solucionar.

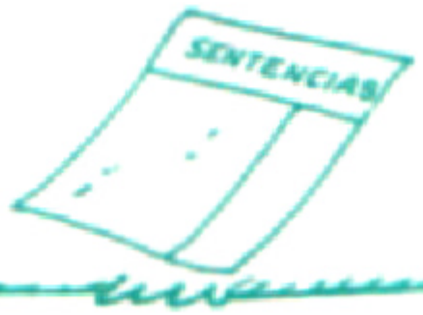


Piensa en todas las aplicaciones diferentes que tienen los ordenadores. El de una planta química lo más probable es que utilice lenguaje FORTRAN, ya que éste es el más indicado para los trabajos de tipo científico.

```
      READ (5,10) (A(I),I=1,N)
10    FORMAT(5F6.2)
      DO 99 I=1,N
      SUM = SUM + A(I)
99    CONTINUE
```

El ordenador de un banco utilizará el lenguaje COBOL, que es el que mejor se presta a las tareas de contabilidad.

```
READ ARCHIVO
IF PAGO-TOTAL IS LESS THAN MAX
     ADD PAGO-EXTRA TO
     PAGO-TOTAL
ADD PAGO-SEMANA TO REGISTRO
CLOSE ARCHIVO
```



¿Y todos los ordenadores tienen que saber todos los lenguajes?

El ordenador no tiene que saber ninguno. ¿Sabes lo que es un intérprete?

Creo que es una persona que escucha lo que dice alguien en una lengua para repetírselo después a otra persona en una lengua distinta.

Eso es. En las Naciones Unidas, por ejemplo, cuando habla el representante de cada país, los intérpretes traducen su discurso a las diferentes lenguas de los otros representantes, y así todo el mundo se entera de lo que está diciendo.

Bueno, pues los ordenadores también tienen un intérprete. Se llama *compilador*.

El compilador es otro tipo de programa que también permanece en la memoria. Su función es leer el programa en la lengua que haya usado el programador y traducírselo al ordenador.

¿Traducirlo a qué?

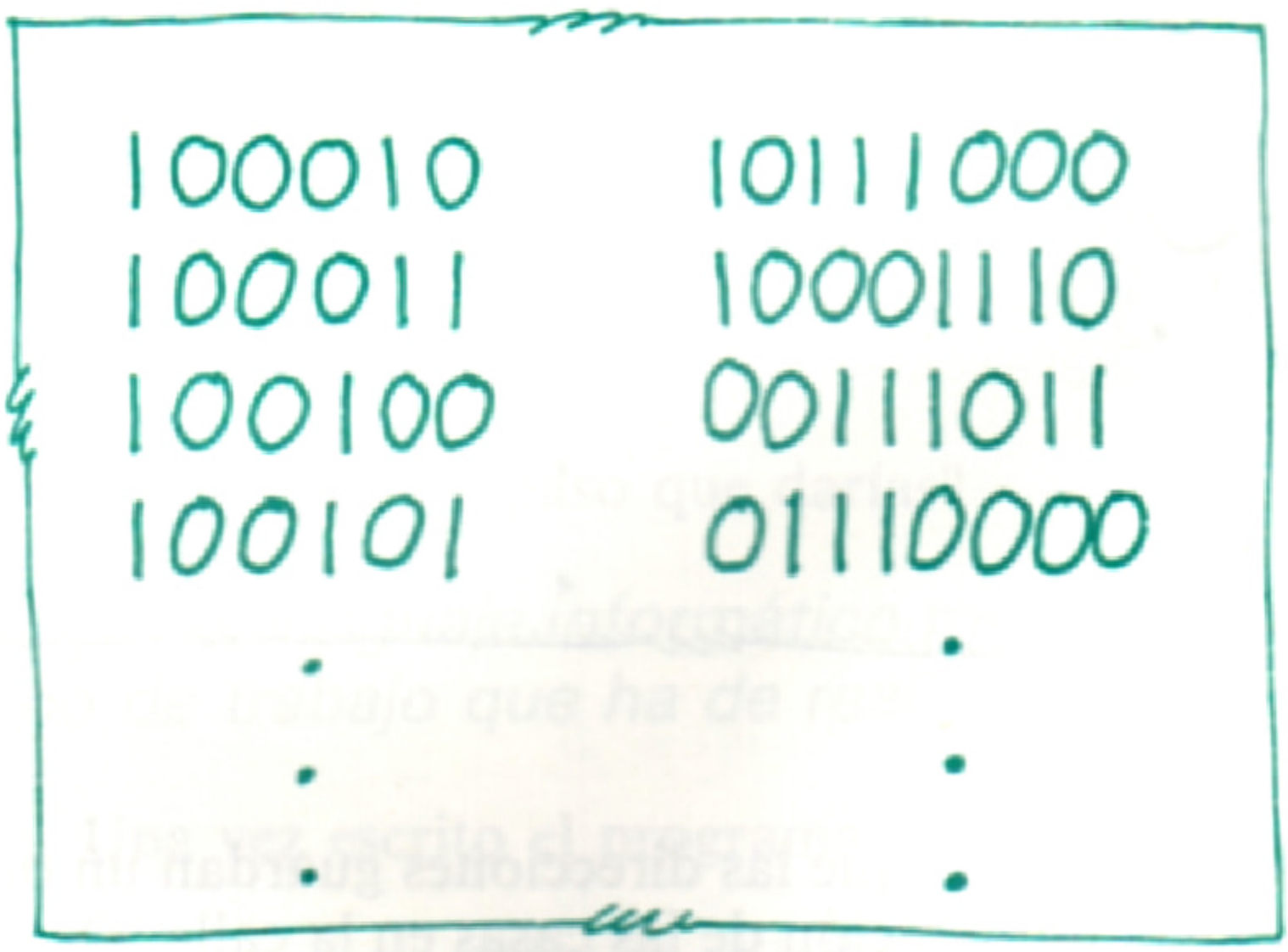
¿No te acuerdas de cómo se almacenaba en el ordenador toda la información?

¡Ah, sí! En un sistema binario.

Exactamente. El programador no puede escribir el programa en términos binarios; por tanto, utiliza uno de los muchos lenguajes informáticos que hemos mencionado.

Luego el compilador lee cada instrucción del programa y la traduce al sistema binario, almacenándola en la memoria del ordenador hasta que la CPU tenga que usarla.

Mira. Una vez compilado en términos binarios y almacenado en la memoria, el programa tendría este aspecto:



Los números de la columna de la derecha son las instrucciones, y los de la izquierda son sus *direcciones*.

¿Direcciones? ¿Como las de las casas?

Sí, lo mismo. Cada lugar de la memoria en el que haya almacenado un número o un carácter tiene una dirección, a fin de que la CPU pueda encontrarlo.

Y como las instrucciones del programa están, asimismo, almacenadas en la memoria, también tienen que tener direcciones.

DIRECCION	INSTRUCCION
34	LEER OTRO NUMERO
35	SUMARLO A TOT
36	SUMAR 1 AL CONTADOR
37	IR AL 22
.	.
.	.
.	.

Observa también que las direcciones guardan un orden, lo mismo que la numeración de las casas en la calle. De este modo, cuando el compilador le dice a la CPU la dirección de la primera instrucción, éste sólo tendrá que añadir 1 para saber la siguiente..., y la otra, y la otra, y la otra, sencillamente añadiendo 1 cada vez.

¡Como el contador del diagrama de flujo!

Igual. Ahora vamos a repasar algunas de las ideas que acabamos de ver, no vaya a ser que las hayas olvidado.

Eres un programador y tienes que escribir un programa, ¿qué es lo que harías en primer lugar?

¿Cuál es el siguiente paso que darías?

Escoger el lenguaje informático más adecuado para el tipo de trabajo que ha de realizar el ordenador.

Bien. Una vez escrito el programa, ¿dónde lo pondrías?

En un disco o una cinta magnética hasta el momento de ejecutarlo.

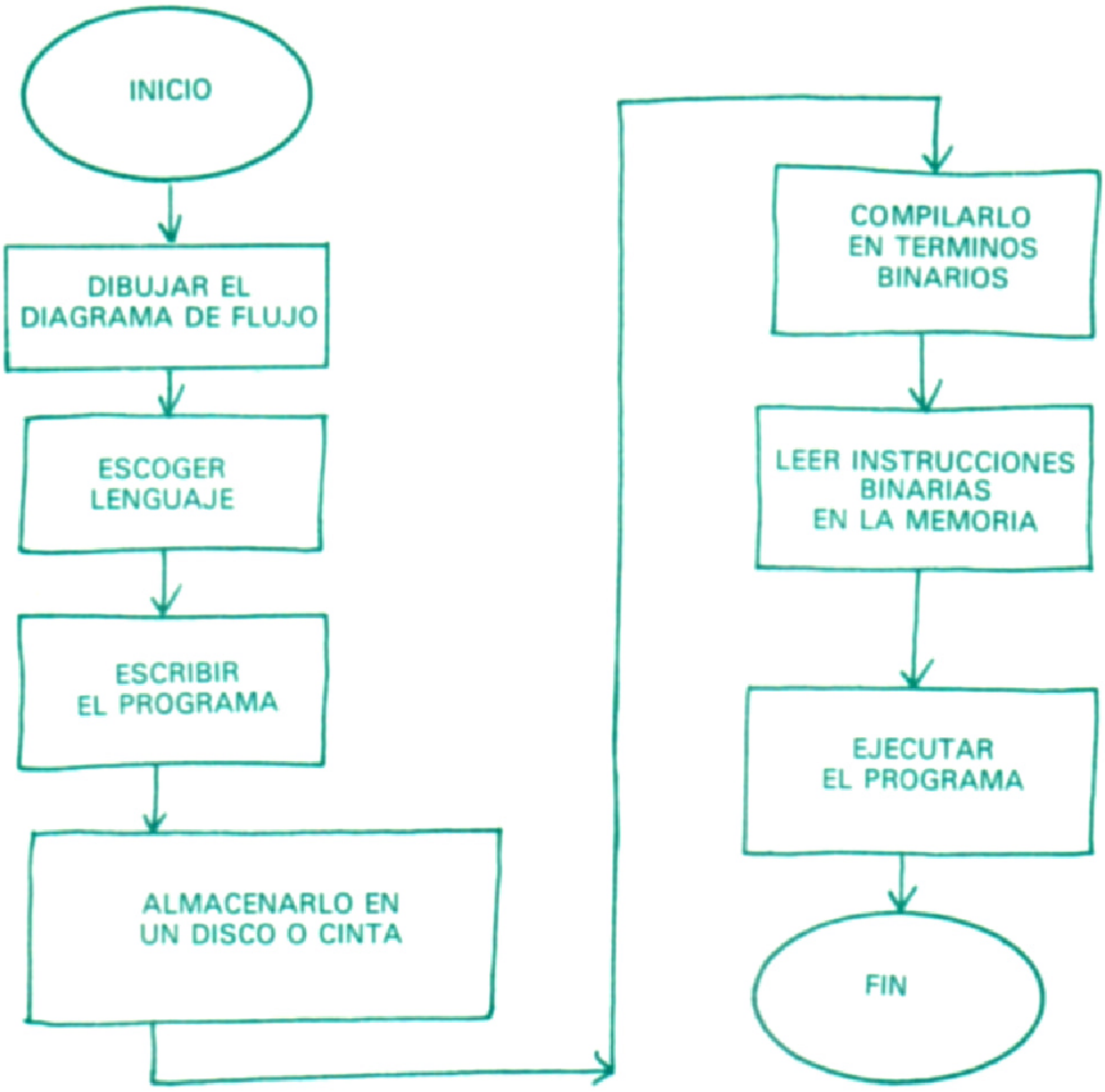
¿Y luego?

Luego el compilador lo traduce a términos binarios y lo copia en la memoria del ordenador.

¿Y cómo encuentra la CPU el programa en la memoria?

El compilador le dice dónde está su dirección.

Muy bien. Incluso podemos hacer un diagrama de flujo de toda la operación:



Los diagramas de flujo parecen sencillos; pero apostaría que escribir los programas no lo es tanto.

Algunos son muy difíciles y llevan mucho tiempo. Sólo los programadores profesionales, que tienen mucha experiencia y conocen casi todos los lenguajes, saben cómo abordarlos.

Pero hay algunos lenguajes que son muy fáciles de aprender y pueden usarse para escribir programas sencillos. El más conocido es el BASIC.

Es el que se usa en la mayoría de los ordenadores domésticos y el que se enseña en las escuelas y colegios que cuentan con ordenadores. Utiliza unas órdenes que son tan fáciles de comprender como de emplear.

Aquí tienes un programa escrito en BASIC, aunque en este caso se trate de uno especialmente sencillo:

```
10  LET A=9
20  LET B=3
30  LET C=11
40  LET D=A+B+C
50  PRINT D
60  END
```

Este programa le dice a la CPU que sume los números 9, 3 y 11 y que imprima la solución. ¿Te acuerdas de cómo llamábamos a los números que procesa la CPU?

Me parece que datos.

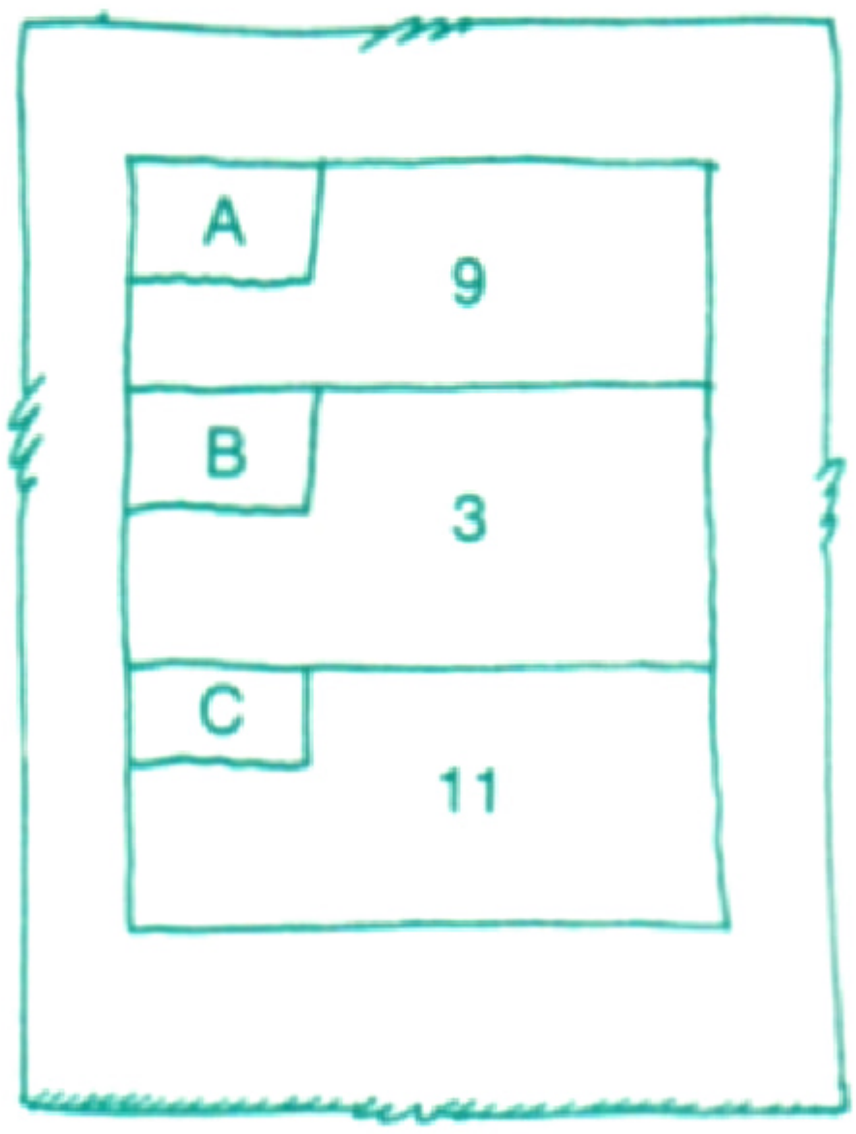
¿Y a la parte de la CPU que suma el valor de los datos?

Sección aritmética.

Bien.

¿Qué significan la A, la B y la C?

Representan aquellas direcciones de la memoria de las que hablábamos antes. Los datos quedarán almacenados en esas direcciones para que la CPU pueda encontrarlos.



El lenguaje BASIC nos permite utilizar letras para las direcciones, lo que resulta muy cómodo, porque son fáciles de recordar. Posteriormente, el compilador cambiará las letras por números.

Pero ahora imagínate que quisiéramos sumar otros tres números, pongamos por ejemplo, 2, 14 y 6. ¿Seguiría sirviendo ese programa?

No, porque los números que éste dice que sumemos son el 9, 3 y 11. Tendríamos que cambiar esos números.

Pero entonces el programa sólo serviría para sumar 2, 14 y 6, y volveríamos a estar en las mismas. ¿Cómo podríamos hacer para que sumara tres números cualesquiera que fuesen sin tener que andar cambiándolos cada vez?

Pues es muy fácil. En lugar de poner los datos en las instrucciones dejamos que las instrucciones lean los datos.

¿Que los lean de dónde? ¡Ah, ya lo sé!, de la entrada.

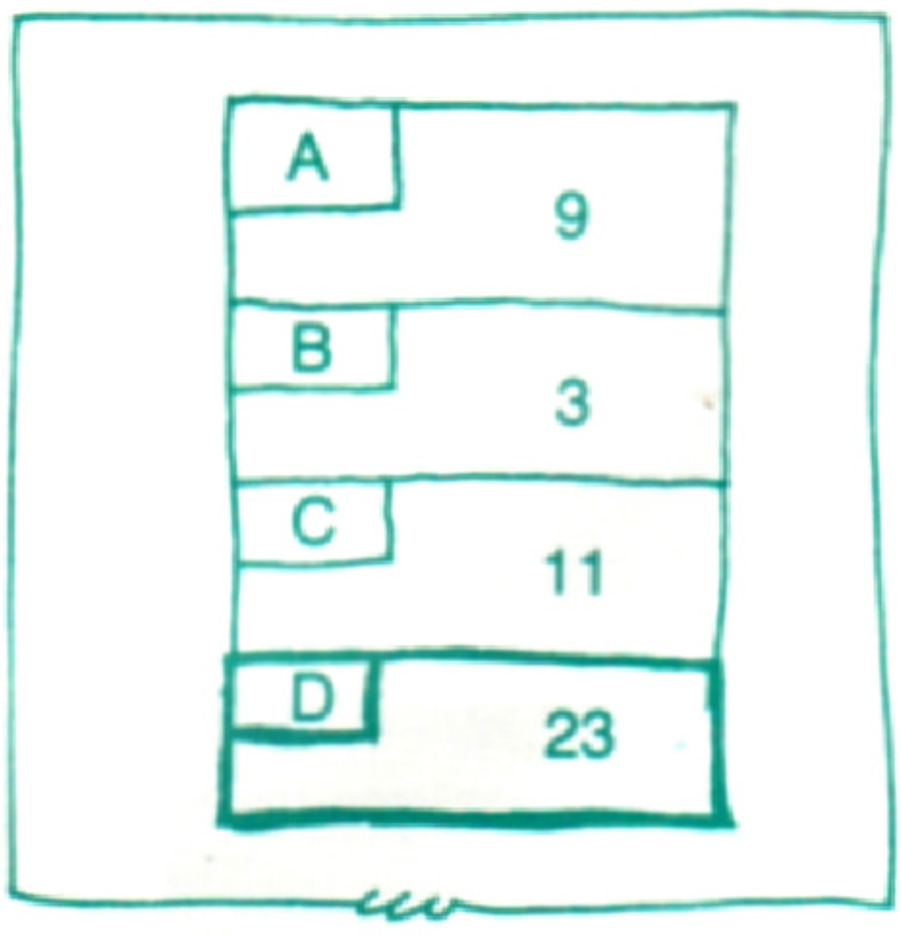
Exactamente. Por tanto, necesitaremos una instrucción que se encargue de hacerlo.

```
10  INPUT A,B,C
20  LET D=A+B+C
30  PRINT D
40  END
```

La instrucción INPUT hará que el programa espere a que alguien teclee los tres números en el teclado del terminal. Luego esos datos serán almacenados en las direcciones A, B y C.

A ver si adivinas tú ahora lo que representa D.

Debe de ser la dirección en la que se almacena el resultado de la suma.



Correcto. Así, la CPU sabrá dónde encontrar la respuesta cuando tenga que imprimirla.

Y me imagino que eso es lo que hace la orden PRINT.

No fallas una. ¿Ves qué fácil es?

Hagamos otro. ¿Qué te parece uno para encontrar la media de edad de mi clase?

Dicho y hecho:

```
10  INPUT A
20  IF A = 0 GO TO 60
30  LET S = S + A
40  LET K = K + 1
50  GO TO 10
60  LET V = S / K
70  PRINT V
80  END
```

¡Pero sólo hay una dirección en la instrucción de la entrada de datos, y las edades que tenemos que sumar son muchas más!

¿Cuántas cucharillas usas para comerte una copa de helado? ¿Una para cada bocado que te lleves a la boca?

¡Claro que no! Uso la misma para todo el helado.

Eso es lo que estamos haciendo aquí: usamos la misma dirección una y otra vez. En lugar de leer todas las edades al mismo tiempo utilizando una dirección diferente para cada una, las leemos de una en una utilizando cada vez la misma dirección.

Después de leer la edad de cada persona en el teclado y de añadirla a la suma, volvemos a la instrucción INPUT y leemos la siguiente, del mismo modo que tú vuelves a llenar la cucharilla para cada bocado.

Me imagino que con eso se ahorra un montón de espacio en la memoria.

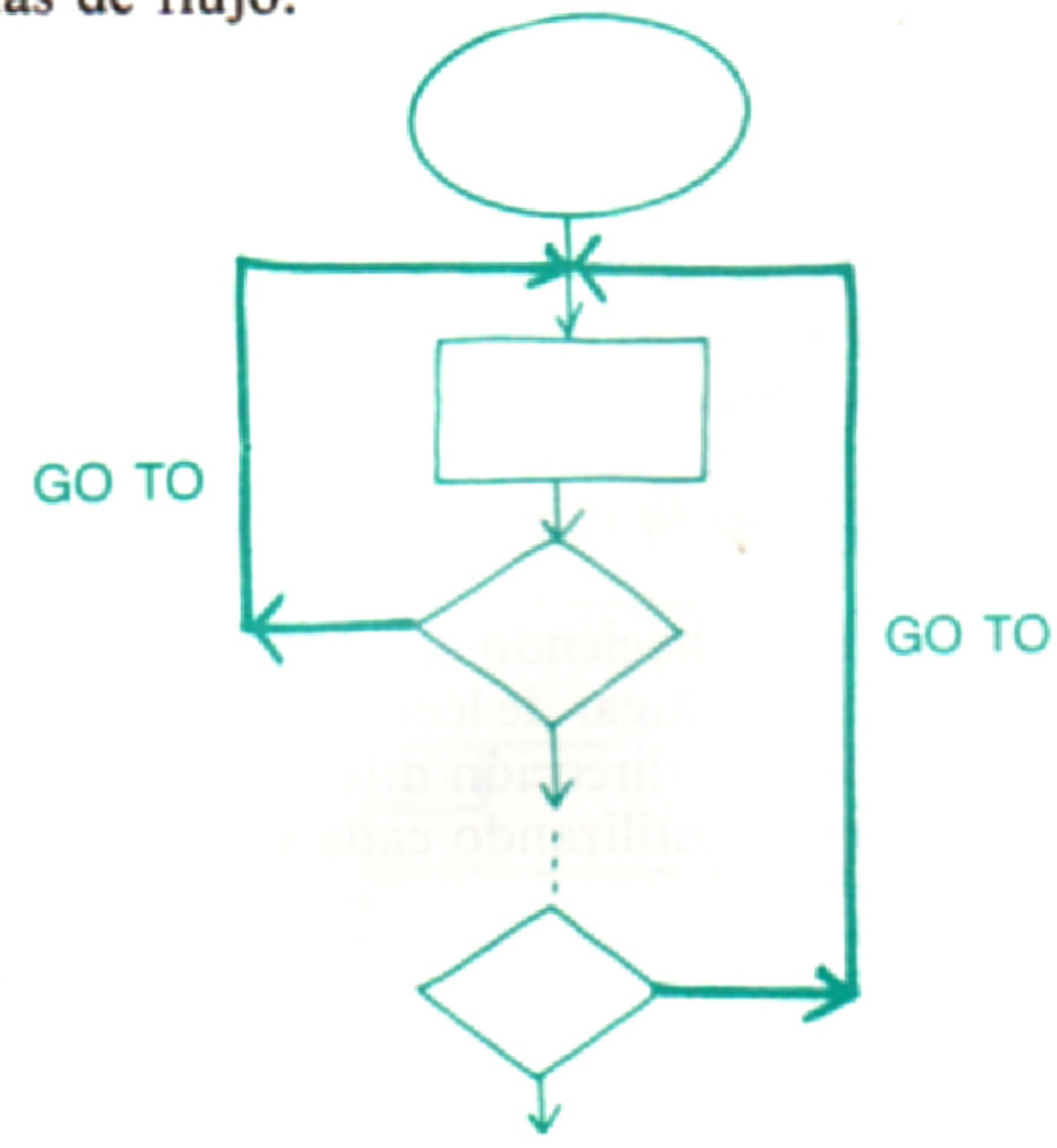
Así es. Y recuerda que en la memoria no hay mucho espacio y no queremos malgastarlo.

¿Por qué pregunta el programa si A es cero? Una persona no puede tener cero años.

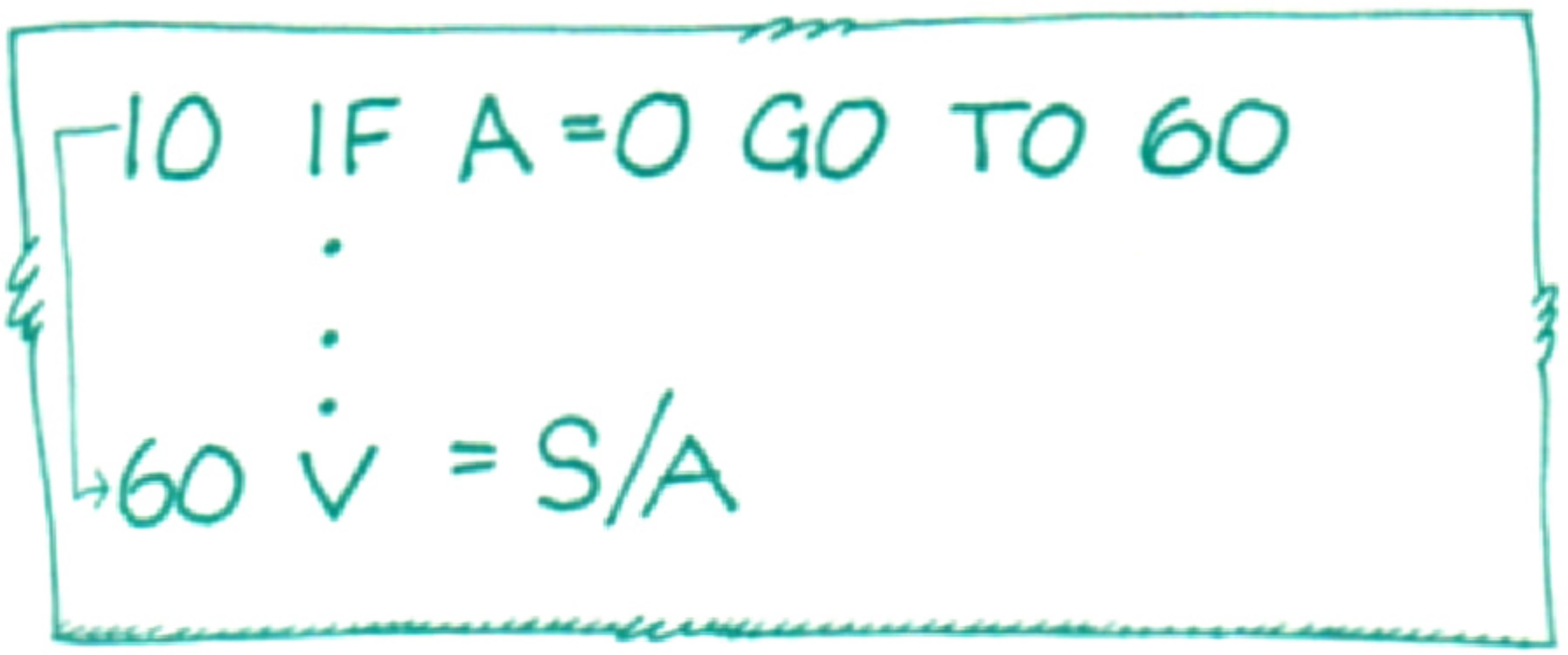
Exactamente. Ponemos ese número para decirle a la CPU que hemos llegado al final de la lista de edades. De no hacerlo, el ordenador se quedaría ahí parado cada vez esperando que alguien le tecleara el siguiente número.

¿Y qué significan esos GO TO que aparecen?

GO TO, en inglés, quiere decir «ve a», y en el lenguaje BASIC equivale a aquellas largas flechas que utilizábamos en los diagramas de flujo.



Recuerda que la CPU va realizando, *por orden,* las instrucciones que se le dan. Pero el primer GO TO dice IF A es igual a cero, no pases a la siguiente instrucción; hemos acabado, así que ve (GO TO) en su lugar a la instrucción número 60.



Y el segundo GO TO le dice que vuelva a repetir las instrucciones comprendidas entre la 10 y la 50, en vez de ir a la instrucción 60.

Entonces es así como retrocedemos para leer la edad de cada persona.

Exactamente. Ese es nuestro bucle.

La instrucción número 30 hace que el ordenador vaya añadiendo a la suma cada nueva edad. Y la 40... ¿sabrías decirme lo que hace?

Esa debe de ser el contador: añade 1 cada vez que se lee una nueva edad.

Y así luego podemos dividir el resultado de la suma por el contador, para saber cuál es el promedio, V, que es lo que dice la instrucción 60.

Y la instrucción 70 le dice que imprima la respuesta.

Eso es. De modo que si has hecho con cuidado el diagrama de flujo y escrito correctamente todas las instrucciones, el programa funcionará siempre. El ordenador te dará siempre la respuesta correcta, independientemente de cuantos datos introduzcas en la entrada.

¿Y qué pasa si he cometido una falta al escribir el programa? ¿Qué hace el ordenador en ese caso?

Todo el mundo comete errores de vez en cuando. Y un error en un programa podría hacer que toda la información que nos diera el ordenador al final fuera inexacta.

Si sucede esto, el programador tiene que buscar el error y corregirlo.

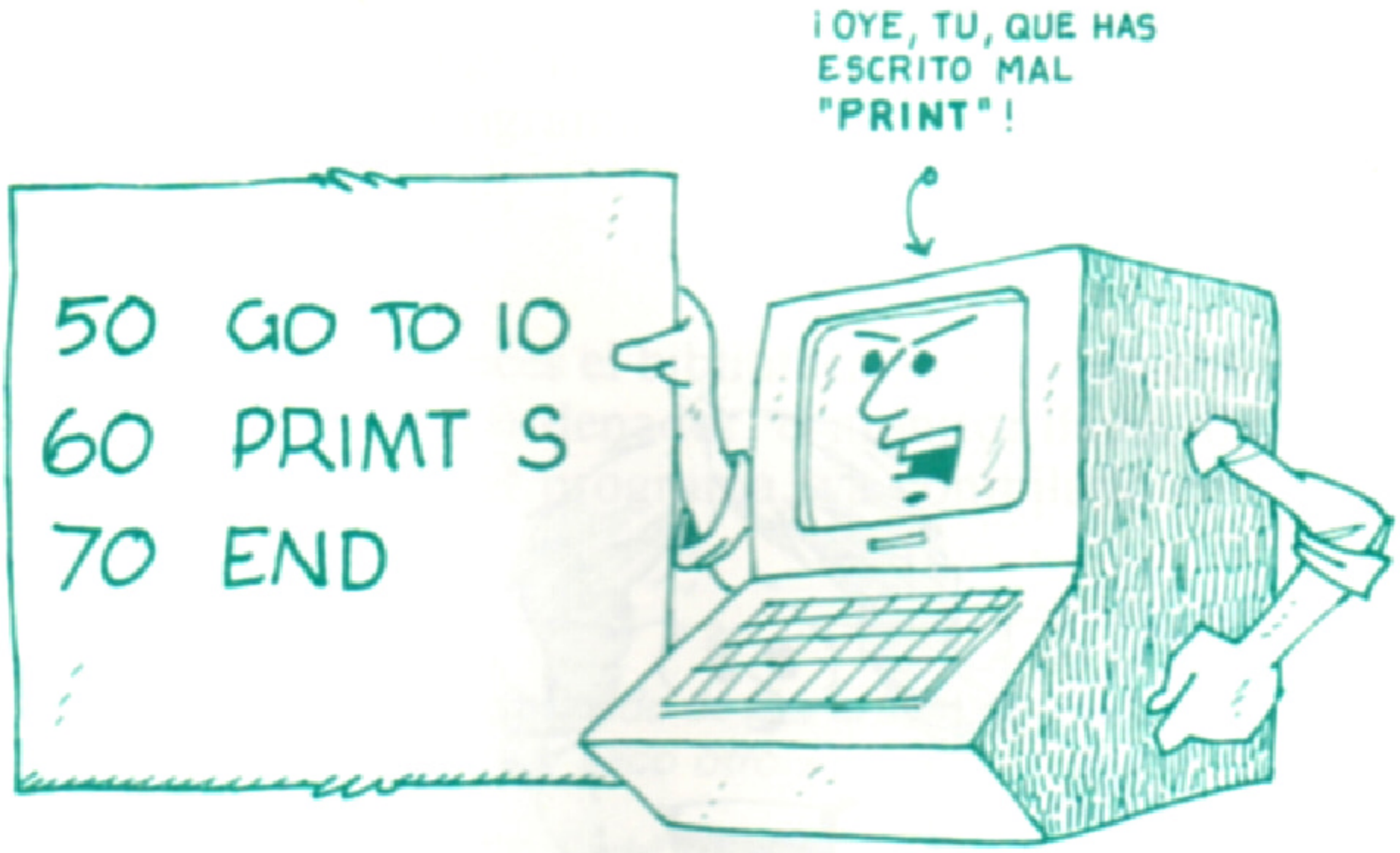
Los errores en los programas se llaman *bugs*.

¿Bugs?

Sí, ése es su nombre. *Bug*, en inglés significa bicho, chinche. Y el proceso de buscar el error y corregirlo se llama depuración o *debugging*. Cuando se trata de programas largos y complicados, este proceso puede convertirse en una tarea difícil y pesada.

Afortunadamente, el ordenador cuenta con mecanismos que ayudan al programador a encontrar los errores del programa. Por ejemplo, el compilador lee detenidamente cada instrucción antes de traducirla a términos binarios. Si contiene una palabra o un símbolo que no reconoce, imprime un mensaje para hacérselo saber al programador.

Por ejemplo, si hubiéramos escrito la palabra PRIMT en el programa BASIC que acabamos de hacer, en lugar de PRINT, el compilador nos lo hubiera dicho antes de continuar.



Y hay otros *programas del sistema* que permiten al programador examinar el código binario en la memoria una vez que el programa ha sido compilado. O ir haciendo una por una todas las instrucciones hasta encontrar el error.

¿A qué programas te refieres cuando dices programas del sistema?

Un programa del sistema es aquel que permanece en la memoria para facilitar la ejecución de los programas que escribimos. El monitor es un programa del sistema. ¿Te acuerdas de cuál era su función?

Sí. Se encarga de que todos los programas ocupen el lugar que les corresponde y de que todos ellos sean ejecutados siguiendo un turno.

Eso es. Y algunos monitores pueden, de hecho, ayudar a corregir el programa.

Pero el monitor hace mucho más que todo eso. Es una especie de bibliotecario de una biblioteca muy concurrida. Si pensamos que el propio ordenador es una biblioteca, en ese caso todos los libros son...

Y entonces, cada vez que alguien escribe un nuevo programa es como cuando la biblioteca adquiere un nuevo libro. Y al igual que el bibliotecario almacena el libro en el estante que le corresponde, el monitor almacena el programa...

En un disco o en una cinta.

Y ejecutar un programa es lo mismo que consultar un libro. Le dices al bibliotecario cuál es el libro que quieres consultar, y él te lo busca, del mismo modo que el monitor busca el programa en un disco o en una cinta.

Luego tienes que registrar el libro para sacarlo de la biblioteca..., así como el programa ha de ser copiado en...

La memoria.

Exactamente. Entonces el bibliotecario pone en el libro un sello con la fecha. En el ordenador, el monitor llama al compilador, y luego entrega el programa, ya compilado en código binario, a la CPU.

Y el bibliotecario lo vuelve a poner en su sitio y te consigue el nuevo; del mismo modo, el monitor sustituye en la memoria el programa utilizado por uno nuevo.

¿Y si tengo que hacer alguna pregunta?

Cuando quieres saber algo, se lo preguntas al bibliotecario; en el ordenador, se lo preguntas al monitor.

En todos los lugares en los que se utilizan ordenadores se puede ver a la gente tecleando en los terminales. Lo que están haciendo es una de estas cuatro operaciones:

introduciendo datos en el ordenador,
pidiendo datos al ordenador,
escribiendo un programa, o
ejecutando un programa.

El monitor se encarga de todas estas tareas.

¿Pero quién escribe el monitor?

Los programadores con mucha experiencia. Al fin y al cabo, el monitor es un programa, y los ordenadores no pueden hacer nada sin programas.

Y como los programas los escriben las personas, eso quiere decir que los ordenadores no pueden hacer nada sin ellas.

Esto es totalmente cierto. Y hablando de escribir, todavía tienes que terminar ese trabajo que estabas haciendo para el colegio. Más vale que pongas manos a la obra cuanto antes.

GLOSARIO

ARCHIVO.—Colección de «registros» que contienen una información similar, como los nombres de los alumnos en un colegio o los precios de los artículos en una tienda.

BASE DE DATOS.—Colección de información sobre una organización o aplicación en la cual cada parte que la compone puede escribirse en la pantalla o modificarse con la misma facilidad con la que se pueden añadir partes nuevas.

BASIC.—Lenguaje de programación muy fácil de aprender y de usar que se encuentra en la mayoría de los ordenadores.

BINARIO.—Sistema numérico, utilizado por los ordenadores, en el cual todos los números están representados por dos símbolos, que son 0 y 1.

BIT.—Un solo 0 ó 1 en una representación binaria de un número o unidad de información.

BUCLE.—Sección de un programa que se repite un número determinado de veces.

BUG.—Error cometido al escribir un programa para ordenador; la búsqueda y corrección del error se llama *debugging*.
BUS.—Camino por el que se envían los datos entre la memoria del ordenador y los puertos de entrada y salida.
BYTE.—Un grupo de 8 bits utilizado para representar en el ordenador un número, una letra o un carácter especial.

CINTA.—Se utiliza cinta magnética como mecanismo de almacenamiento cuando es muy grande la cantidad de información que ha de ser almacenada, ya sea para que el ordenador la utilice posteriormente o para recoger la producida por éste.
COMPILADOR.—Un programa dentro del sistema que traduce otros programas a código binario a fin de que el ordenador pueda entenderlos.
CONTADOR.—Una variable, que generalmente empieza en 0, a la que la CPU añade 1 cada vez que repite una operación.
CONTROL.—La sección de la CPU que lee y lleva a cabo las instrucciones del programa.
CONTROL DE PROCESOS.—El uso de los ordenadores para dirigir la maquinaria o las operaciones en una planta industrial o una fábrica.
CPU.—«CENTRAL PROCESSING UNIT» («UNIDAD CENTRAL DE PROCESO»); la parte del ordenador que se encarga de las operaciones de aritmética, lógica y control.
CURSOR.—Carácter intermitente que aparece en la pantalla del ordenador; puede moverse mediante un mando o un «ratón», para hacer dibujos o seleccionar información en un «menú».

CHIP.—Circuito fabricado en un pequeño trozo de silicio, encapsulado en plástico y con patitas metálicas para realizar las conexiones.

DATOS.—La información utilizada o producida por un programa.
DECIMAL.—Sistema numérico en el que todos los números están representados por uno o más de los siguientes símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

DIAGRAMA DE FLUJO.—Esquema que muestra las operaciones que ha de llevar a cabo un programa.

DIRECCION.—Lugar numerado en la memoria del ordenador en el que se almacenan unidades de información.

DISCO.—Mecanismo que se parece a un disco de música, utilizado para almacenar grandes cantidades de información, ya sea para que el ordenador la utilice posteriormente o la producida por éste.

ENTRADA (INPUT).—La información que se introduce en el ordenador para que la procese el programa.

HARDWARE.—La maquinaria que forma un ordenador.

INPUT.—Véase «ENTRADA».

LAPIZ OPTICO (MANDO DE JUEGO —«JOYSTICK»— y RATON —«MOUSE»—).—Todos ellos son mecanismos para hacer dibujos o leer las direcciones de los puntos en la pantalla del ordenador; los lápices ópticos además pueden utilizarse para leer información impresa codificada, información que envía al ordenador como entrada.

LOGICA.—La sección de la CPU que toma decisiones.

MEMORIA.—Parte del ordenador en la que se almacena la información (datos y programas) mientras es utilizada por la CPU.

MENSAJE.—Frase que es generada por el programa y aparece escrita en la pantalla.

MONITOR.—Programa, generalmente almacenado en la memoria ROM, que supervisa la entrada y salida de la información y controla la ejecución de los programas. (**Nota:** el término «monitor» se suele utilizar también para designar la pantalla del terminal.)

MOUSE («RATON»).—Véase «LAPIZ OPTICO».

OUTPUT.—Véase «SALIDA».

PALABRA.—Unidad básica de información normalmente formada por uno o dos bytes, que a veces puede ser más larga.

PROCESADO.—Operaciones realizadas por el ordenador para proporcionar la información que le pide el usuario.

PROGRAMA.—Conjunto de instrucciones paso a paso que le dicen a la CPU qué es lo que tiene que hacer con la información que le proporciona el usuario.

PROGRAMADOR.—Persona que escribe programas para ser ejecutados por un ordenador.

PROGRAMAS DEL SISTEMA («SYSTEM PROGRAMS»).—Programas que ayudan al usuario a llevar a cabo ciertas tareas de proceso. El monitor y el compilador son ejemplos de programas dentro del sistema.

RAM.—Siglas de «RANDOM ACCESS MEMORY»; la parte de la memoria del ordenador en la que puede almacenarse, para ser utilizada inmediatamente, la información.

RATON («MOUSE»).—Véase «LAPIZ OPTICO».

RED.—Número indeterminado de ordenadores situados en diferentes lugares que pueden intercambiarse información, normalmente mediante las líneas telefónicas o vía satélite.

REGISTRO.—Grupo de datos relacionados que representan una parte independiente de la información, tales como un nombre, una dirección, o el precio de un artículo determinado en una tienda.

ROBOT.—Mecanismo controlado por un ordenador que puede programarse para llevar a cabo trabajos físicos.

ROM.—Siglas de «READ ONLY MEMORY»; la parte de la memoria del ordenador que contiene los programas que no pueden cambiarse.

SALIDA («OUTPUT»).—La información que sale del ordenador, ya sea impresa, por pantalla, almacenada en algún me-

canismo de almacenamiento o es directamente utilizada para el control de alguna función.

SENSOR.—Pequeño mecanismo que sirve para medir temperaturas, presiones u otro tipo de propiedades físicas.

SIMBOLO.—Signo o figura que representa a otra cosa. En un ordenador 0 y 1 son símbolos que pueden representar números o SI y NO.

TERMINAL.—Parte del equipo del ordenador que consta de un teclado y una pantalla y que se utiliza para comunicarse con él.

Los ordenadores son como el cerebro humano: pueden sumar y restar, tomar decisiones y controlar cosas, tienen memoria, pero...,
¿cómo lo hacen?

¡Date una vuelta por el interior de un ordenador y aprende qué es la CPU!

MICROS: BIPS, PITIDOS Y LUCES te llevará dentro de la CPU para que entiendas bien cómo funciona.

Acompañado por tu inseparable amigo, aprenderás cómo cuentan los ordenadores y por qué son capaces de realizar operaciones con tanta rapidez, qué son los lenguajes de ordenador, los programas del sistema y los diagramas de flujo... ¡Y darás tus primeros pasos en BASIC!

ANAYA MULTIMEDIA